The Journal of Open Source Software

SAWNERGY: A Python framework for dynamic residue-interaction networks and walk-based embeddings from molecular dynamics simulations

Yehor Mishchyriak[1], **Sean Stetson**[1, 2], **and Kelly M. Thayer**[1, 2, 3]

1 Department of Mathematics and Computer Science, Wesleyan University, Middletown, CT, United States **2** College of Integrative Sciences, Wesleyan University, Middletown, CT, United States **3** Department of Chemistry, Wesleyan University, Middletown, CT, United States

DOI:

Software

- Review
- Repository
- Archive

Submitted:
Published:

Summary

SAWNERGY is a Python toolkit that turns molecular dynamics (MD) trajectories into temporal residue interaction networks (TRINs), samples random walks, and trains DeepWalk-style skip-gram embeddings. It yields compact low-dimensional representations of residue interaction patterns, replacing bulky framewise adjacencies, and stores them as compressed Zarr archives with metadata for reproducibility and visualization. Here "residue" denotes a biopolymer building block, particularly amino acids in proteins, although the approach can extend to other biomolecules when residue mappings and non-bonded energies are available.

Statement of need

MD simulations yield framewise pairwise interaction data that scales as $O(N^2)$ in the number of residues N, and each residue's interaction vector captures only its immediate neighborhood rather than the broader network context. The combination of high dimensionality and locality makes raw MD data cumbersome for analysis or machine learning. Yet long-range interaction patterns are essential for understanding allosteric[1] effects caused by mutations or ligand binding, which is key in drug design. Hence, we need compact, context-rich representations to make MD-derived features usable. A well-established solution is DeepWalk, a random-walk-based representation learning technique that summarizes multi-hop context in low-dimensional vectors and outperforms linear projections like PCA on graph benchmarks (Perozzi, Al-Rfou, and Skiena 2014). To apply this to residue interaction networks, moving from raw weighted adjacencies to embeddings, one would need to glue together a large multi-stage workflow, which is error-prone, likely to be inefficient, and often lacks reproducible outputs. SAWNERGY adapts the DeepWalk algorithm to weighted residue interaction graphs and packages the full pipeline from MD outputs to embeddings into a light Python framework. It is MD-format agnostic, involves parallel computation, post-run clean-up, visualization and animation capabilities, data compression along with metadata for reproducibility, documentation, and tests.

Interaction model

SAWNERGY focuses on non-bonded interaction energies, namely electrostatic and van der Waals, computed from standard MD force fields (Maier et al. 2015). They follow the

[1] *Allostery is when a change at one site in a protein alters the structure or function at a distant site.*

Coulomb and Lennard–Jones forms, respectively:

$$E_{\text{elec}}(i,j) = \frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}}, \qquad E_{\text{vdW}}(i,j) = 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right].$$

Here, i, j index atoms, q_i and q_j are their partial charges, r_{ij} is the interatomic distance, ε_0 is the vacuum permittivity, and $\varepsilon_{ij}, \sigma_{ij}$ are the Lennard–Jones well depth and size (zero-crossing distance) parameters from the underlying force field. Using these forms, `cpptraj` evaluates pairwise non-bonded energies between atoms in the system, which SAWNERGY then aggregates into attractive and repulsive residue–residue interaction energies.

Why electrostatic and van der Waals interactions?

Electrostatic and van der Waals interaction energies are the dominant non-bonded terms shaping residue–residue communication in folded proteins, and multiple studies from our group have shown that these quantities capture the functional reorganization of allosteric networks under mutation or ligand rescue. In p53 Y220C[2], electrostatic interaction networks differentiate native and mutant substates, reveal long-range communication pathways, and track shifts induced by allosteric effectors (Han, Abramson, and Thayer 2022; Han and Thayer 2024; Cowan and Thayer 2025). Energetic network comparisons also identify residues whose interaction patterns revert toward wild-type upon successful rescue, linking changes in local interaction energies to global structural response (Stetson, Caballero Mancía, and Thayer 2025). Across these studies, electrostatics and van der Waals contributions together provide a sensitive, low-level physical signal from which meaningful RINs can be constructed. Additionally, these terms encode the energetic consequences of common inter-residue contacts, including salt bridges, hydrogen bonds, and packing interactions, since such contacts manifest as characteristic patterns in the underlying Coulomb and Lennard-Jones potentials.

Pipeline description

RIN construction

Given topology and trajectory files and a molecule ID in the system, SAWNERGY calls `cpptraj` (Roe and Cheatham 2013) to compute atomic interaction matrices, parsing EMAP/VMAP blocks, projecting to residues, pruning, symmetrizing, and normalizing to transitions, then writing Zarr archives with metadata.

Let $A \in \mathbb{R}^{n_a \times n_a}$ be the atomic matrix and $P \in \{0,1\}^{n_a \times n_r}$ map atoms to residues. The residue interaction matrix is

$$R = P^\top A P \in \mathbb{R}^{n_r \times n_r}.$$

This projection is needed because `cpptraj`'s `pairwise` driver reports atom–atom energies; residue-level interactions are not emitted directly.

SAWNERGY splits R into attractive and repulsive interaction channels:

$$R^-_{ij} = \max(-R_{ij}, 0), \quad R^+_{ij} = \max(R_{ij}, 0),$$

then prunes by per-row quantile, zeros self-interactions, symmetrizes $\tilde{R} = \frac{1}{2}(R + R^\top)$, and row-normalizes to obtain a transition matrix T with $\sum_j T_{ij} = 1$. Residue centers of mass are recorded for visualization.

Outputs are chunked into Zarr v3 groups and can be compressed to read-only ZIP stores.

[2] *Y220C mutates a tyrosine to cysteine in the p53 DNA-binding domain, destabilizing the protein and impairing tumor-suppressor function.*

For embeddings we recommend using the attractive channel, because stabilizing contacts (hydrogen bonds, salt bridges, hydrophobic packing) hold the fold together and define the meaningful co-occurrence structure along walks, whereas repulsive contributions are transient exclusions that add noise without encoding the binding network.

Walk sampling

Given a transition matrix T and length L, we treat residues as states in a Markov process and draw walk sequences $v_{0:L}$ from T, starting at each residue in turn and recording the visited nodes. Transition stacks are loaded into shared memory so parallel workers sample without copies, and the sampler cleans up shared segments when done.

Self-avoiding walks enforce no node revisits. SAWNERGY lets users mix in a fraction of SAWs (`saw_frac`) alongside plain random walks. This trade-off loosely mirrors node2vec's p, q biases (Grover and Leskovec 2016): plain walks revisit neighborhoods (BFS-like), while higher `saw_frac` encourages exploration of more distant regions of the graph like DFS.

Embedding

SAWNERGY trains skip-gram (full softmax) or SGNS (skip-gram with negative sampling) models over the sequences of random walk visits to predict pairs of co-occurring residues.

For SGNS, given a true pair (u, v) from a walk sample and set of random pairs $\mathcal{N}$ sampled from a distribution proportional to frequency counts across all the walks, we use gradient descent to minimize the following loss

$$\mathcal{L}_{\text{SGNS}} = -\left[\log \sigma(\mathbf{u}^\top \mathbf{v}) + \sum_{n \in \mathcal{N}} \log \sigma(-\mathbf{u}^\top \mathbf{n})\right],$$

where σ is the logistic sigmoid. SGNS learns to distinguish true co-occurrences from sampled noise, implicitly learning embeddings $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$.

Full-softmax skip-gram learns $P(\text{context} \mid \text{target})$ over the entire vocabulary for each target node, also learning embeddings $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$. The model minimizes

$$\mathcal{L}_{\text{SG}} = -\sum_{(u,v)} \log \frac{\exp(\mathbf{u}^\top \mathbf{v})}{\sum_{w \in V} \exp(\mathbf{u}^\top \mathbf{w})}.$$

Both objectives yield compact vectors that encode interaction context.

For cross-frame comparisons, SAWNERGY includes an orthogonal alignment helper (`align_frames`) that solves the Procrustes problem

$$\min_{R \in O(d)} \|XR - Y\|_F \quad \Rightarrow \quad R^* = UV^\top \text{ for SVD}(X^\top Y) = U\Sigma V^\top,$$

with optional centering and reflection control, enabling post-hoc alignment of embeddings from different frames or runs, where X and Y are embedding matrices and R^* is the optimal orthogonal transformation aligning X to Y.

Training backends include PureML (NumPy) (Mishchyriak 2025; Harris et al. 2020) and optional PyTorch (Paszke et al. 2019). Per-frame embeddings are stored in the same compressed Zarr format (Miles et al. 2025) with metadata; RINs and embeddings are visualized via Matplotlib-based components (Hunter 2007) in `sawnergy.visual.Visualizer` and `sawnergy.embedding.Visualizer`. For temporal consistency and faster convergence, training warm-starts each frame from the embedding of the previous frame before further optimization.

Note: these steps are performed for every frame or batch of frames, with in-batch interaction averaging during TRIN construction, specified via `frame_batch_size`*.*

Quality control

The GitHub repository includes a `tests/` suite invoked via `pytest` covering storage helpers, walk sampling, embedding utilities, and math helpers. These tests run in continuous integration on each commit to the public repository and before PyPI releases to ensure reproducibility and stability. SAWNERGY is actively used within ThayerLab, including ongoing analyses of the 12 known p53 isoforms.

Example usage

0. Configure logging:

```
from sawnergy.logging_util import configure_logging
import logging
configure_logging("logs", console_level=logging.INFO, file_level=logging.ERROR)
```

1. Build a RIN archive from an MD trajectory:

```
from sawnergy.rin import RINBuilder
RINBuilder().build_rin(
    topology_file="topo.prmtop",
    trajectory_file="traj.nc",
    molecule_of_interest=1, # which molecule ID to process
    frame_batch_size=10, # frames per batch for averaging
    prune_low_energies_frac=0.85, # drop lowest 85% per row
    include_attractive=True, # write attractive channel
    include_repulsive=False, # skip repulsive channel
    num_matrices_in_compressed_blocks=10, # matrices per compressed archive block
    compression_level=3, # Blosc compression level (3/9)
    output_path="RIN.zip"
)
```

2. Sample random and self-avoiding walks:

```
from sawnergy.walks import Walker
with Walker("RIN.zip") as w:
  w.sample_walks(
      walk_length=20, # steps per walk
      walks_per_node=100, # walks per residue
      saw_frac=0.25, # fraction of walks that are self-avoiding
      include_attractive=True, # use attractive interactions
      include_repulsive=False, # skip repulsive interactions
      in_parallel=False, # sample serially (not multi-process)
      output_path="WALKS.zip"
  )
```

3. Train embeddings (DeepWalk-style skip-gram/SGNS):

```
from sawnergy.embedding import Embedder
emb = Embedder("WALKS.zip")
emb.embed_all(
    RIN_type="attr", # choose attractive RIN
    using="merged", # merge plain and self-avoiding walks
    num_epochs=5, # training epochs
    negative_sampling=True, # use SGNS
```

```
    window_size=5, # context window for co-occurrence
    num_negative_samples=10, # fake samples per true sample
    dimensionality=128, # embedding dimension
    model_base="pureml", # backend ('pureml' or 'torch')
    shuffle_data=True, # shuffle training pairs
    kind="in", # return embeddings from the first layer of the model
    output_path="EMBEDDINGS.zip"
)
```

4. Visualize embedding of the first frame/frame-batch:

```
from sawnergy.embedding import Visualizer
v = Visualizer("EMBEDDINGS.zip", normalize_rows=True)
v.build_frame(1, show=True)
```

5. Visualize the RIN:

```
from sawnergy.visual import Visualizer
v = Visualizer("RIN.zip", antialiased=True)
v.build_frame(1, node_colors="rainbow", show=True)
```

Visual example produced by SAWNERGY

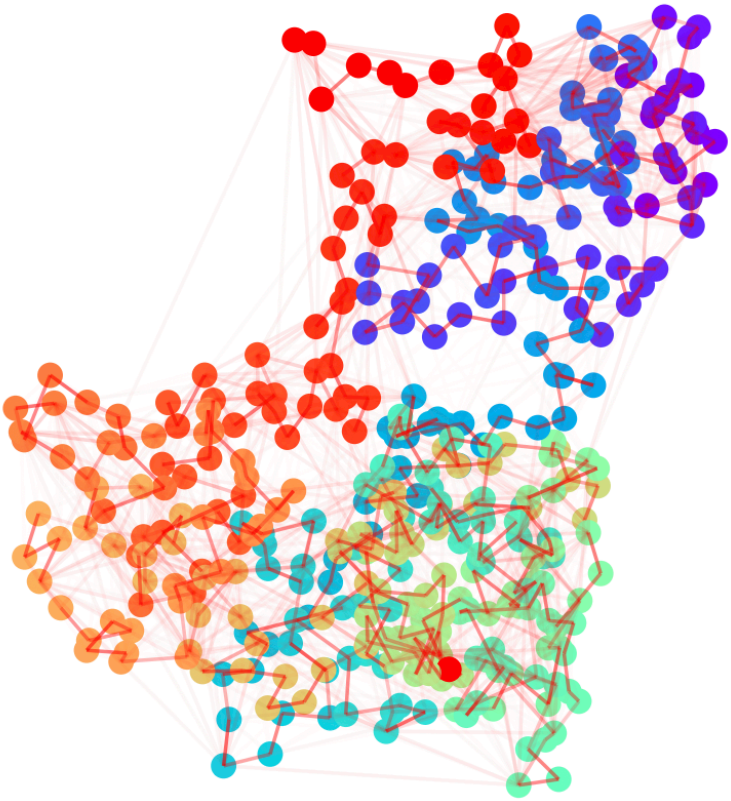

Figure 1: RIN of p53 Tumor Suppressor Protein

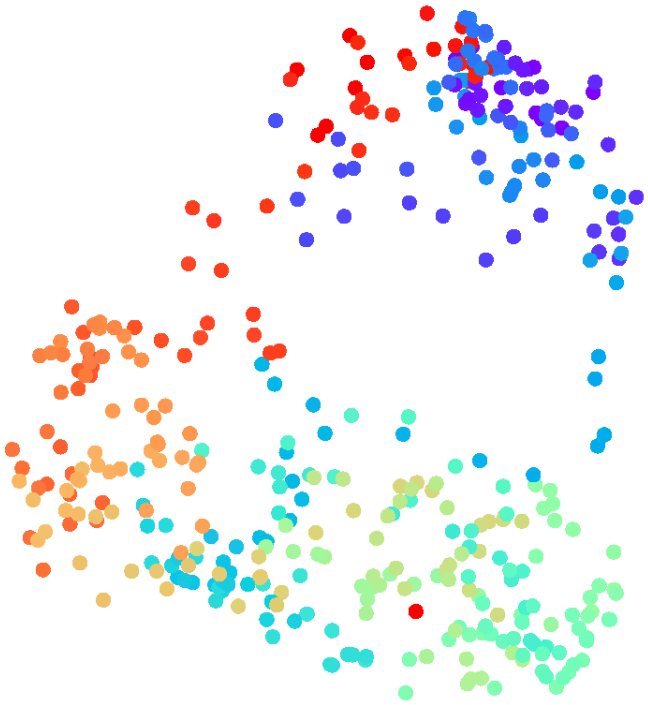

Figure 2: Embedding of p53 Tumor Suppressor Protein Projected onto 3 leading PCs

Note the visual resemblance: the first plot uses residue centers of mass, while the second plot is derived purely from random walks on the energetic network.

Potential applications

- Feature engineering: use framewise embeddings as inputs to ML models for stability, binding, or mutational effects.
- Dynamics: cluster or reduce per-frame embeddings to map states, transitions, and rare events.
- Comparative analysis: align embeddings across trajectories/conditions to quantify perturbations (mutations, ligands, pH/temperature shifts).

Availability

Source code: https://github.com/Yehor-Mishchyriak/SAWNERGY
PyPI: https://pypi.org/project/sawnergy/
Documentation: https://ymishchyriak.com/docs/SAWNERGY-DOCS
License: Apache-2.0 (see `LICENSE`)

Acknowledgements

We thank the Molecules to Medicine consortium for fruitful discussions, especially Dr. David L. Beveridge and Dr. Michael P. Weir and their lab members. We are

grateful to Henk Meij for technical support with Wesleyan's high-performance computing resources. This work was supported by NIH R15 GM128102-02, NSF CHE-2320718, and NSF CNS-0619508/CNS-095985 (Wesleyan HPC facilities).

References

Cowan, Benjamin S., and Kelly M. Thayer. 2025. "Network Theory Analysis of Allosteric Drug-Rescue Mechanisms in the Tumor Suppressor Protein P53 Y220C Mutant." *International Journal of Molecular Sciences* 26 (14): 6884. https://doi.org/10.3390/ijms26146884.

Grover, Aditya, and Jure Leskovec. 2016. "Node2vec: Scalable Feature Learning for Networks." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 855–64. ACM. https://doi.org/10.1145/2939672.2939754.

Han, In Sub M., Dylan Abramson, and Kelly M. Thayer. 2022. "Insights into Rational Design of a New Class of Allosteric Effectors with Molecular Dynamics Markov State Models and Network Theory." *ACS Omega* 7 (3): 2831–41. https://doi.org/10.1021/acsomega.1c05624.

Han, In Sub M., and Kelly M. Thayer. 2024. "Reconnaissance of Allostery via the Restoration of Native P53 DNA-Binding Domain Dynamics in Y220C Mutant P53 Tumor Suppressor Protein." *ACS Omega* 9 (18): 19837–47. https://doi.org/10.1021/acsomega.3c08509.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. "Array Programming with NumPy." *Nature* 585: 357–62. https://doi.org/10.1038/s41586-020-2649-2.

Hunter, John D. 2007. "Matplotlib: A 2D Graphics Environment." *Computing in Science & Engineering* 9 (3): 90–95. https://doi.org/10.1109/MCSE.2007.55.

Maier, James A., Carlos Martinez, Koushik Kasavajhala, Lauren Wickstrom, Kevin E. Hauser, and Carlos Simmerling. 2015. "ff14SB: Improving the Accuracy of Protein Side Chain and Backbone Parameters from ff99SB." *Journal of Chemical Theory and Computation* 11 (8): 3696–3713. https://doi.org/10.1021/acs.jctc.5b00255.

Miles, Alistair, John Kirkham, David Stansby, Dimitri Papadopoulos Orfanos, Joe Hamman, et al. 2025. "Zarr-Developers/Zarr-Python: V3.1.5." https://doi.org/10.5281/zenodo.17672242.

Mishchyriak, Yehor. 2025. "PureML: A Lightweight, NumPy-Based Deep Learning Framework." https://github.com/Yehor-Mishchyriak/PureML.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, et al. 2019. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." *Advances in Neural Information Processing Systems* 32. https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. 2014. "DeepWalk: Online Learning of Social Representations." In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 701–10. ACM. https://doi.org/10.1145/2623330.2623732.

Roe, Daniel R., and Thomas E. Cheatham. 2013. "PTRAJ and CPPTRAJ: Software for Processing and Analysis of Molecular Dynamics Trajectory Data." *Journal of Chemical Theory and Computation* 9 (7): 3084–95. https://doi.org/10.1021/ct400341p.

Stetson, Sean, Fernando Caballero Mancía, and Kelly M. Thayer. 2025. "Restoration of the Y220C P53 Full-Length Mutant by PK11000: A Molecular Dynamics Study of an Intrinsically Disordered Protein." *ACS Omega*. https://doi.org/10.1021/acsomega.5c05639.