Graphein - a Python Library for Geometric Deep Learning and Network Analysis on Protein Structures

Arian R. Jamasb\textsuperscript{1, 2} Pietro Li\`o\textsuperscript{2} Tom L. Blundell\textsuperscript{1}

Abstract
Graphein is a python library for constructing graph and surface-mesh representations of protein structures for computational analysis. The library interfaces with popular geometric deep learning libraries: DGL, PyTorch Geometric and PyTorch3D. Geometric deep learning is emerging as a popular methodology in computational structural biology. As feature engineering is a vital step in a machine learning project, the library is designed to be highly flexible, allowing the user to parameterise the graph construction, scalable to facilitate working with large protein complexes, and containing useful pre-processing tools for preparing experimental structure files. Graphein is also designed to facilitate network-based and graph-theoretic analyses of protein structures in a high-throughput manner. As example workflows, we make available two new protein structure-related datasets, previously unused by the geometric deep learning community.

Availability and implementation: Graphein is written in python. Source code, example usage and datasets, and documentation are made freely available under a MIT License at the following URL: https://github.com/a-r-j/graphein

Introduction
Geometric deep learning refers to the application of deep learning methods to data with an underlying non-Euclidean structure, such as graphs or manifolds (Bronstein et al., 2016). These methods have already been applied to a number of problems within computational biology, and indeed computational structural biology (Fout et al., 2017; Torng & Altman, 2019; Gligorijevic et al., 2019; Uhl et al., 2019; Zamora-Resendiz & Crivelli, 2019; Sanyal et al., 2020; Gainza et al., 2019). Geometric deep learning libraries have emerged, providing graph representation functionality and in-built datasets - typically with a focus on small molecules (Wang et al., 2019; Fey & Lenssen, 2019). Featurisation schemes and computational analysis of molecular graphs are a mature area of research within cheminformatics. However, data preparation for geometric deep learning in structural biology is yet to receive the same attention. Protein structures are significantly more complex than small molecules, and so greater control over the data engineering and featurisation process is required.

Proteins form complex three dimensional structures to carry out cellular functions. Decades of structural biology research, have resulted in a large pool of experimentally-determined protein structures. However, it is not clear how best to represent these data in machine learning experiments. 3DCNNs applied to grid-structured representations of protein structures and sequence-based methods have proved commonplace (Ragoza et al., 2017; Sato & Ishida, 2019; Pu et al., 2019). However, these representations fail to capture relational information in the context of intramolecular contacts and the internal chemistry of the biomolecular structures. Furthermore, these methods can suffer from difficulties in their application to datasets with variable input sizes and 3DCNNs are computationally inefficient due to convolving over large regions of empty space, often requiring experimenters to restrict the volume of the protein to regions of interest, thereby losing global structural information. For instance, in the case of protein-ligand interaction and binding affinity prediction, this often takes the form of restricting the volume to be centred on a binding pocket, thereby losing information about allosteric sites on the protein and possible conformational rearrangements that contribute to the binding process. Furthermore, 3D volumetric representations are not translationally and rotationally invariant, therefore these datasets often require augmentation to overcome this. In the case of biased datasets, that do not accurately represent the entirety of protein space, this can plausibly limit generality. Graphs suffer relatively less from these problems as they are translationally and rotationally
invariant. Structural descriptors of position can be leveraged in the case of molecules with chiral centres. Graph representations can enable relatively more efficient computation than 3DCNN methods.

Proteins can very naturally be represented as graphs, at various spatial scales. Residue-level graphs represent protein structures as graphs where the nodes consist of amino-acid residues and the edges the relations between them - often based on intramolecular interactions or euclidean distance-based cutoffs. Atom-level graphs represent the protein structure in a manner consistent with small-molecule graph representations, where nodes represent individual atoms and edges the relations between them - often chemical bonds or, again, distance-based cutoffs. The graph structure can further be elaborated by assigning numerical features to corresponding nodes and edges. These features can represent, for instance, chemical properties of the residue or atom-type, secondary structure assignments or solvent accessibility metrics of the residue. Edge features can include bond or interaction types, or distances.

Graph representations of proteins have been successfully used in machine learning and structural analysis projects in structural biology (Pires et al., 2011; 2013; Cheng et al., 2008). Web-servers for computing protein structure graphs exist, however the lack of public APIs for programmatic access, limited featurisation schemes and incompatibility with deep learning libraries motivated the development of Graphein (Chakrabarty et al., 2019; Vijayabaskar et al., 2011).

**Graphein**

Graphein consists of two core classes for making protein structure graphs and surface meshes respectively. Structure graphs are compatible with DGL (Wang et al., 2019), PyTorch Geometric (Fey & Lenssen, 2019) and NetworkX (Hagberg et al., 2008), and surface meshes are compatible with PyTorch3D (Ravi et al., 2020). To our knowledge, this is the first application of PyTorch3D for protein structure data. Example visualisations of graph and mesh construction are provided in Figure 1; an overview of mesh and graph construction, and node and edge featurisation schemes are given in Figure 2.

**Protein Structure Graphs**

**NODE REPRESENTATIONS**

Graphs can be constructed for all chains contained within a polypeptide structure, or for a user-defined selection of chains. This is useful in the contexts where regions of interest on a protein may be localised to a single chain. For residue-level graphs, users can choose between an atom-based residue representation (e.g. \( \alpha \)-carbon or \( \beta \)-carbon), or sidechain centroid. Sidechain centroids are calculated as the centre of gravity of the deprotonated residue. Functionality for featurising nodes is provided in Graphein. Features for a DGL graph are stored as a dictionary of PyTorch tensors attached to each node. Residue-level graph nodes can be featurised using low-dimensional embeddings of amino acid physico-chemical properties from Kidera et al. (Kidera et al., 1985) and Meiler et al (Meiler et al., 2001) or a one-hot encoding of amino acid type. In addition, functionality for including a one-hot encoded representation of eight state secondary structure and solvent accessibility metrics (ASA, RSA, SSA) calculations from DSSP (Kabsch \& Sander, 1983) are provided. XYZ positions are also added as node features. Functionality for user-defined node or edge features is also provided.

**EDGE REPRESENTATIONS**

Functionality for computing intramolecular graph edges is provided through GetContacts (GetContacts). Euclidean distance-based edges can be computed with a user-defined threshold. Functionality for constructing \( k \)-nearest neighbour graphs, where two vertices are connected by an edge if they are among the \( k \) nearest neighbours by Euclidean distance is included. Graph edges can also be added on the basis of Delaunay triangulation. Delaunay triangles correspond to joining points that share a face in the 3D Voronoi diagram of the protein structures. Edge featurisation for atom-level graphs is provided through the feature schemes available in DGL.Chem, which depend on RDKit (Landrum et al., 2020). All of these edge types can be included in the same multirelational graph; as these different edge representations capture varying aspects of structural information, this could be usable in a setting where different components of a model operate on each of these graphs. A Long Interaction Network (LIN) parameter controls the minimum required separation in the amino acid sequence for edge creation. This can be useful in reducing the number of noisy edges under distance-based edge creation schemes.

**Protein Structure Meshes**

The protein structure mesh class consists of a wrapper for PyMOL and Pytorch3D (Schrödinger, LLC, 2015). PyMol is used to produce a .Obj file from either a PDB accession code or a provided .pdb file. The .Obj file is passed to Pytorch3D to produce a tensor-based representation of the protein surface as vertices and faces. The user can specify a number of parameters controlling the surface calculation to PyMol, and thus the final mesh. These parameters include specifying solvent inclusion, solvent probe radius, surface mode (\{triangles,dots\}), surface quality (resolution of mesh).
Datasets

As examples, we make available two graph-based protein structure datasets. The first, based on the collections outlined in (Zeng et al., 2019), consists of 420 proteins, with node labels indicating whether a residue is involved in a protein-protein interaction. The interaction status data and structure originate from structures of the complexes in the RCSB PDB. The authors make available a set of additional node features based on Position-Specific Scoring Matrices (PSSMs), providing evolutionary context as protein-protein interaction sites are typically conserved, which can be incorporated with the structural node features calculated by Graphein.

The second dataset, based on Protein Structural Change Database (PSCDB) (Amemiya et al., 2011), consists of 904 paired examples of bound and unbound protein structures that undergo 7 classes of conformational rearrangement motion. Two tasks can be formulated with this dataset. The first is the graph classification task of predicting the type of motion a protein undergoes upon ligand binding, the second is an edge prediction task between the paired bound and unbound protein structure graphs. These tasks provide utility in improving understanding of protein structural dynamics in drug development, where molecules are typically docked into largely rigid structures with limited flexibility in the binding pockets in high-throughput in silico screens.

Usage

Example usage and workflows are provided in the documentation at this HTTPS URL: https://www.github.com/a-r-j/graphein.

Conclusion

Geometric deep learning has shown promise in computational biology and structural biology. However, the availability of processed datasets is poor. Graphein is a python
library designed to facilitate construction of datasets for geometric deep learning on proteins. In addition, we make available two datasets for protein-protein interaction site prediction (node classification) and protein conformational rearrangement prediction (graph classification). We hope that graphein serves to further interest in the field and reduce friction in processing protein structure data for geometric deep learning. The library also provides utility in preparing protein structure graphs for graph-theoretic analyses.

Acknowledgements

We thank Paul Scherer, Cristian Bodnar and Ben Day for useful feedback.

References


Sato, R. and Ishida, T. Protein model accuracy estimation based on local structure quality assessment us-


