Machine Learning in Bioinformatics

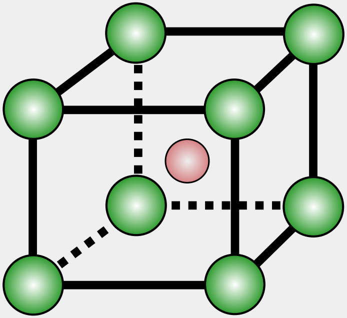
Graph Neural Networks

Philipp Benner
philipp.benner@bam.de

VP.1 - eScience
Federal Institute of Materials Research and Testing (BAM)

February 8, 2026

Graph data



Outline

- Graph Convolutional Neural Networks (GCNN)
- General graph neural networks (GNN)
- Graph isomorphisms and discriminative power of GNNs
- Advanced models and applications

Graph Convolutional Neural Networks (GCNNs)

Graph convolutions

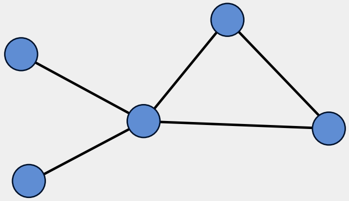
- Convolutions are not only restricted to image and time-series data
- Graph convolutions are used to model the interaction between nodes
- Let $G = (N, E)$ denote a graph with nodes N and edges E
- How could we implement a convolution of G with a weight matrix W?
- The result of a convolution is again a graph[1], i.e.

$$G' = G * W$$

[1]Remember that convolution on images also returns an image

Graph convolutions

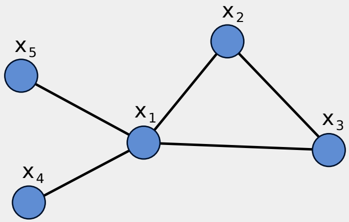
- Graph G with 5 nodes and 5 edges:



- We assign a feature vector $x_i \in \mathbb{R}^p$ to the i-th node
- The feature vector can depend on the type of the node
- Nodes of the same type might share the same feature vector

Graph convolutions

- Graph G with 5 nodes and 5 edges:



- We assign a feature vector $x_i \in \mathbb{R}^p$ to the i-th node
- The feature vector can depend on the type of the node
- Nodes of the same type might share the same feature vector

Graph convolutions

- Let $A = (a_{ij})_{ij} \in \mathbb{R}^{k \times k}$ denote the adjacency matrix of a graph with k nodes
- The strength of the connection between node i and j is given by a_{ij}
- Self-connections $a_{ii} \neq 0$ allow to incorporate the features of the nodes itself
- The convolution operation updates the feature vector of node i by summing over all neighbor nodes, i.e.

$$x_i' = \sigma\left(\sum_j a_{ij} W x_j\right) = \sigma\left(\sum_{j \to i} W x_j\right)$$

where $W \in \mathbb{R}^{p \times p}$ and σ is the activation function[2]

[2]Graph convolutions are *permutation equivariant*

Graph convolutions

- For the full graph we obtain

$$\underbrace{X'}_{k\times p} = \sigma(\underbrace{A}_{k\times k}\underbrace{X}_{k\times p}\underbrace{W^\top}_{p\times p})$$

  where $X \in \mathbb{R}^{k\times p}$ is the matrix of k feature vectors
- Note that the weight matrix W does not depend on the size and connectivity of the graph
- W can be applied to multiple graphs and optimized during training of the graph convolutional neural network (GCNN)
- GCNNs typically apply multiple convolutions and afterwards compute summary statistics of the feature vectors, the result can then be used in a conventional neural network

[2] Many extensions and generalizations exist [Battaglia et al., 2018, Dwivedi et al., 2020]

GCNN
Generalizated Update Rules

Graph convolutions - Self-connections

- Graph convolutional networks as introduced so far, can be efficiently computed, but are limited in their expressive power
- The same weight matrix W is used for all nodes
- A simple extension is to introduce a separate weight matrix V for self-connections

$$x_i' = \sigma\left(Vx_i + \sum_{j \to i} Wx_j\right)$$

- Note that now the sum over $\{j \to i\}$ should not include any self-connections

Graph convolutions - Edge gates

- Another important generalization are edge gates [Marcheggiani and Titov, 2017]
- Edge gates allow the network to learn what edges are important for the graph learning task
- The update function is given by

$$x_i' = \sigma\left(\sum_{j \to i} \eta_{ij} \odot W x_j\right)$$

  where $\odot$ denotes the element-wise multiplication (Hadamard product)
- The $\eta_{ij} \in \mathbb{R}^p$ act as edge gates and are computed as

$$\eta_{ij} = \sigma\left(A x_i + B x_j\right)$$

Graph convolutions - Edge features

- Even more general are networks that contain separate features on edges [Joshi et al., 2019]
- Node features and edge features e_{ij} between nodes i and j are updated as follows

$$x'_i = \sigma\left(\sum_{j \to i} \eta_{ij} \odot W x_j\right)$$

$$e'_{ij} = \sigma\left(A x_i + B x_j + C e_{ij}\right)$$

$$\eta_{ij} = \frac{\sigma(e_{ij})}{\sum_k \sigma(e_{ik}) + \epsilon}$$

- Note that η_{ij} is a normalized version of $\sigma(e_{ij})$

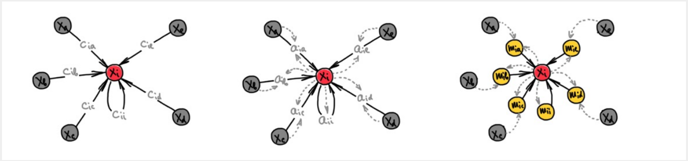
Graph Neural Networks (GNNs)

Permutation equivariance on graphs

- Let $X \in \mathbb{R}^{k \times p}$ be the feature matrix of a graph G with k nodes
- Let $\varphi_G(X)$ denote the result of applying a graph neural network φ_G to X
- $\tau(G, X)$ denotes a row-permutation of X with corresponding relabeling of nodes in G
- We require that φ_G is *equivariant* with respect to τ (permutation equivariant), i.e.

$$\begin{tikzcd}
(G, X) \arrow[r, "\varphi_G"] \arrow[d, "\tau"'] & (G, Y) \arrow[d, "\tau"] \\
(G', X') \arrow[r, "\varphi_{G'}"] & (G', Y')
\end{tikzcd}$$

Graph neural networks (GNNs)



- Three types of GNNs:

  Convolution, Attention, Message passing

Update rule of GNNs [Bronstein et al., 2021]

$$x_i' = \phi\left(x_i, \bigoplus_{j \to i} \psi(x_i, x_j)\right)$$

where $\bigoplus$ is a permutation invariant aggregation function, ϕ and ψ are learnable functions

Graph neural networks - GCNNs

- General update formula of GNNs [Bronstein et al., 2021]:

$$x_i' = \phi\left(x_i, \bigoplus_{j \to i} \psi(x_i, x_j)\right)$$

  where $\bigoplus$ is a permutation invariant aggregation function

- GCNNs are an instance of GNNs, because

$$x_i' = \phi\left(x_i, \bigoplus_{j \to i} \psi(x_i, x_j)\right) = \sigma\left(Vx_i + \sum_{j \to i} Wx_j\right)$$

  where $\phi(x_i, z) = \sigma(Vx_i + z)$, $\bigoplus = \sum$, and $\psi(x_i, x_j) = Wx_j$

Graph neural networks - Self-attention

- General update formula of GNNs [Bronstein et al., 2021]:

$$x_i' = \phi\left(x_i, \bigoplus_{j \to i} \psi(x_i, x_j)\right)$$

  where $\bigoplus$ is a permutation invariant aggregation function

- Self-attention layers are characterized by

$$\psi(x_i, x_j) = a(x_i, x_j)\psi'(x_j)$$

  where a denotes the attention mechanism. The computation of the attention value differs from the attention used in transformers [Veličković et al., 2017]

- The message $\psi'(x_j)$ from node j is weighted by the attention value $a(x_i^\top, x_j)$

Graph neural networks - Message passing

- General update formula of GNNs [Bronstein et al., 2021]:

$$x_i' = \phi\left(x_i, \bigoplus_{j\rightarrow i} \psi(x_i, x_j)\right)$$

  where $\bigoplus$ is a permutation invariant aggregation function

- The most general version of GNNs are *message passing* networks, where ψ is a neural network itself. The message received by node *i* is computed from both the feature vectors of node *i* and *j* [Gilmer et al., 2017]

Disciminative Power of GNNs

Graph isomorphism

Graph isomorphism

Let G and H be two graphs with vertex sets $V(G)$ and $V(H)$. A graph isomorphism f between G and H is defined as a function $f : V(G) \mapsto V(H)$ such that for all vertices u, v adjacent in G it follows that $f(u)$ and $f(v)$ are adjacent in H.

- Two graphs are called isomorphic if there exists a graph isomorphism
- Finding graph isomorphisms is difficult, i.e. for a graph with n nodes we have to test $n!$ node permutations
- *Weisfeiler-Lehman Isomorphism Test* can be used as a computationally fast heuristic

Weisfeiler-Lehman Isomorphism Test (1-WL) [Weisfeiler and Leman, 1968]

For both graphs G and H, assign each node i an initial node color $x_i = 1$. Within each iteration, the node color is updated using a given hash function according to the update rule

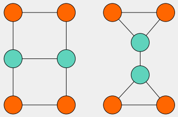
$$x_i \leftarrow \mathrm{hash}\left(x_i, \{\{\, x_j \mid j \to i \,\}\}\right) ,$$

where $\{\{\, \cdot \,\}\}$ denotes a multiset. The hash function maps the current node color and the multiset of neighboring node colors to a new node color from a *discrete* set.

Nodes are partitioned according to their colors. The algorithm terminates if node partitions are stable. G and H pass the test if $n_x(G) = n_x(H)$ for all x, where $n_x(G) = \sum_{i \in G} \mathbb{1}_{x = x_i}$ is the number of occurrences of color x [Huang and Villar, 2021].

Graph isomorphism

- Note that the feature vectors x_i might never become stable, however, the node partitions will
- The Weisfeiler-Lehman Isomorphism Test has limited power
  - { Test fails } $\rightarrow$ graphs are not isomorphic
  - Some graphs that pass the test are not isomorphic
- Example of non-isomorphic graphs that pass the test:



Graph isomorphism - k-WL test

- The k-WL test improves on this difficulty by coloring node sets of size k [Maron et al., 2019]
- The sub-graph structure of nodes in the k-tuples determine the initial k-tuple colors
- Colors are updated based on the colors of the neighborhood of k-tuples, which is all tuples where one node has been replaced by another node
- 1-WL and 2-WL test are equivalent in discriminative power
- Otherwise $k+1$-WL is more powerful than k-WL

Expressive power of graph networks

- Recall the update rule of graph neural networks

$$x_i' = \phi\left(x_i, \bigoplus_{j \to i} \psi(x_i, x_j)\right)$$

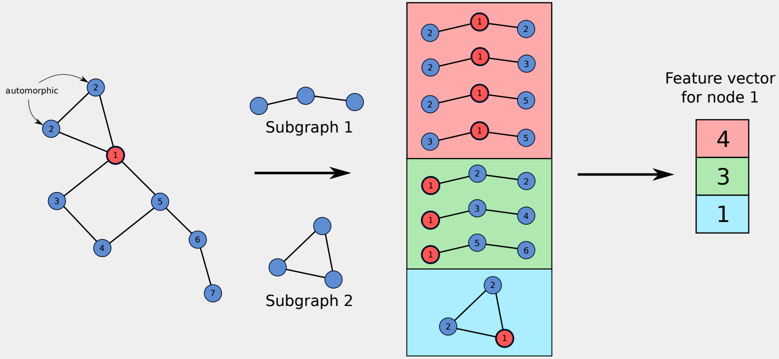
- The rule is identical to the hash function of the 1-WL test
- GNNs are therefore only as powerful as the 1-WL test in discriminating graphs [Xu et al., 2018]
- Although we seem to need permutation equivariance for graph neural networks, we then loose essential information on the graph structure and cannot distinguish between all graphs

Beyond 1-WL

- There exist several strategies to increase the power of GNNs beyond 1-WL:
  - Design models equivalent to the k-WL test with $k > 1$ [Maron et al., 2018, Maron et al., 2019, Keriven and Peyré, 2019, Azizian and Lelarge, 2020]
  - Specific pre-coloring of nodes to encode positional information[3]
    - Pre-coloring based on graph substructures [Bouritsas et al., 2022]
    - Using simplicial- or cell-complexes [Bodnar et al., 2021b, Bodnar et al., 2021a]
    - Graph Laplacian eigenvectors [Dwivedi et al., 2020]
  - Work with sub-graphs and local equivariance, e.g. natural graph networks [de Haan et al., 2020]

[3]Remember that the WL test uses the same initial color for all nodes

Subgraph isomorphic counting



- Features encode local environment through subgraph counting [Bouritsas et al., 2022]

Natural Graph Networks (NGNs)

- Recall the update function of CGNNs

$$x_i' = \sigma\left(\sum_{j\to i} W x_j\right)$$

- NGNs [de Haan et al., 2020] generalize this update formula as follows

$$x_i' = \sigma\left(\sum_{j\to i} W_{ij}^{G} x_j\right)$$

  i.e. the weight matrix depends on the graph G and the edge (i, j).

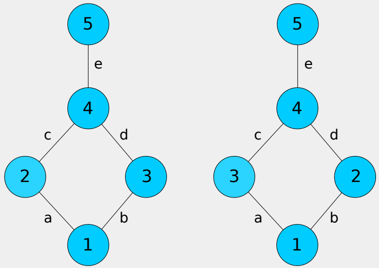
- Isomorphic graphs share the same weights

- Automorphic graphs constrain the weight matrices

Natural Graph Networks (NGNs)

- Automorphic graphs constrain weight matrices
- Given the following graph *G* and assume for simplicity that $W^G_{ij} = W^G_{ji}$

	1	2	3	4	5
1		a	b		
2	a			c	
3	b			d	
4		c	d		e
5				e	



	1	2	3	4	5
1		b	a		
2	b			d	
3	a			c	
4		d	c		e
5				e	

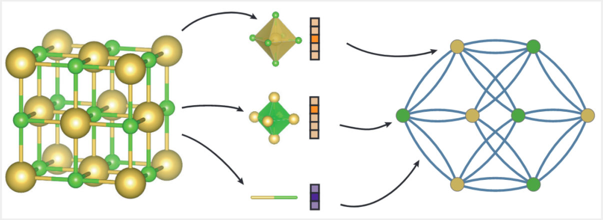
- Edges a, b and c, d must have the same weights, i.e. $W^G_{12} = W^G_{13}$ and $W^G_{24} = W^G_{34}$

Natural Graph Networks (NGNs)

- Natural graphs in this form are too general, i.e. each set of isomorphic graphs receives its own weights
- With this minimal weight sharing no learning across graphs is possible
- *Local natural graphs (LNGs)* solve this issue by looking at small sub-graphs

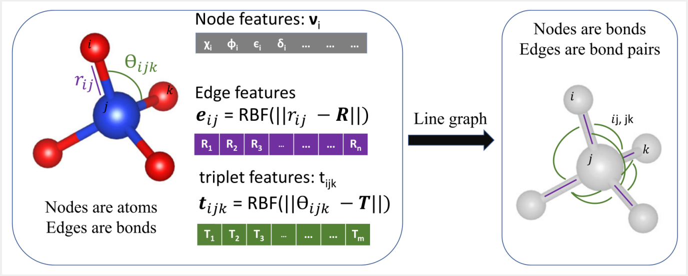
GNNs in Practice

CGCNN



- Crystal Graph Convolutional Neural Network (CGCNN) [Xie and Grossman, 2018], one of the first and simplest crystal graph networks
- Initial node features: Group number, periodic number, electronegativity, covalent radius, valence electrons, first ionization energy, electron affinity, block, atomic volume
- Initial edge features: Atom distance

ALIGNN



Node features: $\mathbf{v}_i$
χ_i ϕ_i ϵ_i δ_i … … …
Edge features
$\boldsymbol{e}_{ij}$ = RBF($||r_{ij} - \boldsymbol{R}||$)
R_1 R_2 R_3 … … … R_n
triplet features: t_{ijk}
$\boldsymbol{t}_{ijk}$ = RBF($||\Theta_{ijk} - \boldsymbol{T}||$)
T_1 T_2 T_3 … … … T_m
r_{ij}
Θ_{ijk}
i
j
k
Nodes are atoms
Edges are bonds
Line graph
Nodes are bonds
Edges are bond pairs
ij, jk

[Choudhary and DeCost, 2021]

ALIGNN

- ALIGNN [Choudhary and DeCost, 2021] performs edge-gated graph convolution simultaneously on both the atomistic bond graph and the line graph
- Atomistic bond graph: Atoms are nodes, bonds are edges
  - Initial node features: Electronegativity, group number, covalent radius, valence electrons, first ionization energy, electron affinity, block, and atomic volume
  - Initial edge features: RBF expanded interatomic bond distances
- Line graph: Bonds are nodes, bond pairs with one common atom (or atom triplets) are edges. Nodes correspond to bonds in the atomistic bond graph
  - Node features: Edge features of the bond graph
  - Initial edge features: RBF expanded bond angles

Furhter reading

- Review of graph neural networks [Zhou et al., 2020]
- Geometric deep learning [Bronstein et al., 2021]

Software

GNNs are implemented in PyTorch Geometric:

- Website:
  `https://pyg.org/`
- Documentation:
  `https://pytorch-geometric.readthedocs.io`

References I

Azizian, W. and Lelarge, M. (2020).
Expressive power of invariant and equivariant graph neural networks.
arXiv preprint arXiv:2006.15646.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018).
Relational inductive biases, deep learning, and graph networks.
arXiv preprint arXiv:1806.01261.

Bodnar, C., Frasca, F., Otter, N., Wang, Y., Lio, P., Montufar, G. F., and Bronstein, M. (2021a).
Weisfeiler and lehman go cellular: Cw networks.
Advances in Neural Information Processing Systems, 34:2625–2640.

References II

Bodnar, C., Frasca, F., Wang, Y., Otter, N., Montufar, G. F., Lio, P., and Bronstein, M. (2021b).
Weisfeiler and lehman go topological: Message passing simplicial networks.
In *International Conference on Machine Learning*, pages 1026–1037. PMLR.

Bouritsas, G., Frasca, F., Zafeiriou, S. P., and Bronstein, M. (2022).
Improving graph neural network expressivity via subgraph isomorphism counting.
IEEE Transactions on Pattern Analysis and Machine Intelligence.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. (2021).
Geometric deep learning: Grids, groups, graphs, geodesics, and gauges.
arXiv preprint arXiv:2104.13478.

References III

Choudhary, K. and DeCost, B. (2021).
Atomistic line graph neural network for improved materials property predictions.
npj Computational Materials, 7(1):1–8.

de Haan, P., Cohen, T. S., and Welling, M. (2020).
Natural graph networks.
Advances in Neural Information Processing Systems, 33:3636–3646.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. (2020).
Benchmarking graph neural networks.
arXiv preprint arXiv:2003.00982.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017).
Neural message passing for quantum chemistry.
In *International conference on machine learning*, pages 1263–1272. PMLR.

References IV

Huang, N. T. and Villar, S. (2021).
A short tutorial on the Weisfeiler-Lehman test and its variants.
In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8533–8537. IEEE.

Joshi, C. K., Laurent, T., and Bresson, X. (2019).
An efficient graph convolutional network technique for the travelling salesman problem.
arXiv preprint arXiv:1906.01227.

Keriven, N. and Peyré, G. (2019).
Universal invariant and equivariant graph neural networks.
Advances in Neural Information Processing Systems, 32.

Marcheggiani, D. and Titov, I. (2017).
Encoding sentences with graph convolutional networks for semantic role labeling.
arXiv preprint arXiv:1703.04826.

References V

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. (2019).
Provably powerful graph networks.
Advances in neural information processing systems, 32.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. (2018).
Invariant and equivariant graph networks.
arXiv preprint arXiv:1812.09902.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017).
Graph attention networks.
arXiv preprint arXiv:1710.10903.

Weisfeiler, B. and Leman, A. (1968).
The reduction of a graph to canonical form and the algebra which appears therein.
NTI, Series, 2(9):12–16.

References VI

Xie, T. and Grossman, J. C. (2018).
Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties.
Physical review letters, 120(14):145301.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018).
How powerful are graph neural networks?
arXiv preprint arXiv:1810.00826.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020).
Graph neural networks: A review of methods and applications.
AI Open, 1:57–81.