WIKIPEDIA
The Free Encyclopedia

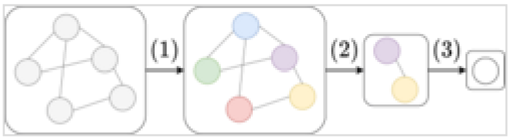
Graph neural network

Graph neural networks (**GNN**) are specialized artificial neural networks that are designed for tasks whose inputs are graphs.[1][2][3][4][5]

One prominent example is molecular drug design.[6][7][8] Each input sample is a graph representation of a molecule, where atoms form the nodes and chemical bonds between atoms form the edges. In addition to the graph representation, the input also includes known chemical properties for each of the atoms. Dataset samples may thus differ in length, reflecting the varying numbers of atoms in molecules, and the varying number of bonds between them. The task is to predict the efficacy of a given molecule for a specific medical application, like eliminating *E. coli* bacteria.

The key design element of GNNs is the use of *pairwise message passing*, such that graph nodes iteratively update their representations by exchanging information with their neighbors. Several GNN architectures have been proposed,[2][3][9][10][11] which implement different flavors of message passing,[12][13] started by recursive[2] or convolutional constructive[3] approaches. As of 2022, it is an open question whether it is possible to define GNN architectures "going beyond" message passing, or instead every GNN can be built on message passing over suitably defined graphs.[14]

In the more general subject of "geometric deep learning", certain existing neural network architectures can be interpreted as GNNs operating on suitably defined graphs.[12] A convolutional neural network layer, in the context of computer vision, can be considered a GNN applied to graphs whose nodes are pixels and only adjacent pixels are connected by edges in the graph. A transformer layer, in natural language processing, can be considered a GNN applied to complete graphs whose nodes are words or tokens in a passage of natural language text.



Basic building blocks of a graph neural network (GNN). (1) Permutation equivariant layer. (2) Local pooling layer. (3) Global pooling (or readout) layer. Colors indicate features.

Relevant application domains for GNNs include natural language processing,[15] social networks,[16] citation networks,[17] molecular biology,[18] chemistry,[19][20] physics[21] and NP-hard combinatorial optimization problems.[22]

Open source libraries implementing GNNs include PyTorch Geometric[23] (PyTorch), TensorFlow GNN[24] (TensorFlow), Deep Graph Library[25] (framework agnostic), jraph[26] (Google JAX), and GraphNeuralNetworks.jl[27]/GeometricFlux.jl[28] (Julia, Flux).

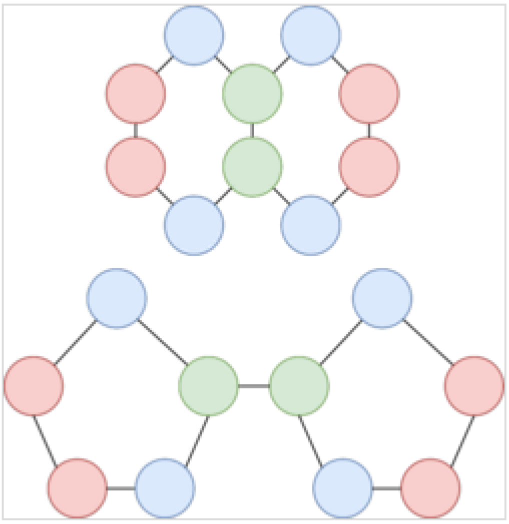
Architecture

The architecture of a generic GNN implements the following fundamental layers:[12]

1. *Permutation equivariant*: a permutation equivariant layer maps a representation of a graph into an updated representation of the same graph. In the literature, permutation equivariant layers are implemented via pairwise message passing between graph nodes.[12][14] Intuitively, in a message passing layer, nodes *update* their representations by *aggregating* the *messages* received from their immediate neighbours. As such, each message passing layer increases the receptive field of the GNN by one hop.
2. *Local pooling*: a local pooling layer coarsens the graph via downsampling. Local pooling is used to increase the receptive field of a GNN, in a similar fashion to pooling layers in convolutional neural networks. Examples include k-nearest neighbours pooling, top-k pooling,[29] and self-attention pooling.[30]
3. *Global pooling*: a global pooling layer, also known as *readout* layer, provides fixed-size representation of the whole graph. The global pooling layer must be permutation invariant, such that permutations in the ordering of graph nodes and edges do not alter the final output.[31] Examples include element-wise sum, mean or maximum.

It has been demonstrated that GNNs cannot be more expressive than the Weisfeiler–Leman Graph Isomorphism Test.[32][33] In practice, this means that there exist different graph structures (e.g., molecules with the same atoms but different bonds) that cannot be distinguished by GNNs. More powerful GNNs operating on higher-dimension geometries such as simplicial complexes can be designed.[34][35][13] As of 2022, whether or not future architectures will overcome the message passing primitive is an open research question.[14]

Message passing layers


Non-isomorphic graphs that cannot be distinguished by a GNN due to the limitations of the Weisfeiler-Lehman Graph Isomorphism Test. Colors indicate node features.

Message passing layers are permutation-equivariant layers mapping a graph into an updated representation of the same graph. Formally, they can be expressed as message passing neural networks (MPNNs).[12]

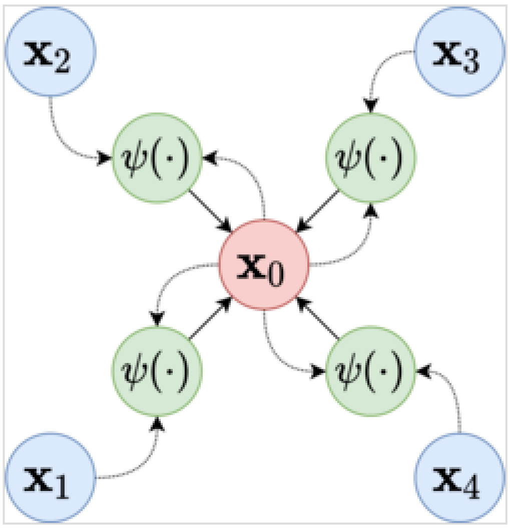
Let $G = (V, E)$ be a graph, where V is the node set and E is the edge set. Let N_u be the neighbourhood of some node $u \in V$. Additionally, let $\mathbf{x}_u$ be the features of node $u \in V$, and $\mathbf{e}_{uv}$ be the features of edge $(u, v) \in E$. An MPNN layer can be expressed as follows:[12]

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in N_u} \psi(\mathbf{x}_u, \mathbf{x}_v, \mathbf{e}_{uv})\right)$$

where ϕ and ψ are differentiable functions (e.g., artificial neural networks), and $\bigoplus$ is a permutation invariant aggregation operator that can accept an arbitrary number of inputs (e.g., element-wise sum, mean, or max). In particular, ϕ and ψ are referred to as *update* and *message* functions, respectively. Intuitively, in an MPNN computational block, graph nodes *update* their representations by *aggregating* the *messages* received from their neighbours.

The outputs of one or more MPNN layers are node representations $\mathbf{h}_u$ for each node $u \in V$ in the graph. Node representations can be employed for any downstream task, such as node/graph classification or edge prediction.

Graph nodes in an MPNN update their representation aggregating information from their immediate neighbours. As such, stacking n MPNN layers means that one node will be able to communicate with nodes that are at most n "hops" away. In principle, to ensure that every node receives information from every other node, one would need to stack a number of MPNN layers equal to the graph diameter. However, stacking many MPNN layers may cause issues such as oversmoothing[36] and oversquashing.[37] Oversmoothing refers to the issue of node representations becoming indistinguishable. Oversquashing refers to the bottleneck that is created by squeezing long-range dependencies into fixed-size representations. Countermeasures such as skip connections[10][38] (as in residual neural networks), gated update rules[39] and jumping knowledge[40] can mitigate oversmoothing. Modifying the final layer to be a fully-adjacent layer, i.e., by considering the graph as a complete graph, can mitigate oversquashing in problems where long-range dependencies are required.[37]



Node representation update in a Message Passing Neural Network (MPNN) layer. Node $\mathbf{x}_0$ receives messages sent by all of its immediate neighbours $\mathbf{x}_1$ to $\mathbf{x}_4$. Messages are computing via the message function ψ, which accounts for the features of both senders and receiver.

Other "flavours" of MPNN have been developed in the literature,[12] such as graph convolutional networks[9] and graph attention networks,[11] whose definitions can be expressed in terms of the MPNN formalism.

Graph convolutional network

The graph convolutional network (GCN) was first introduced by Thomas Kipf and Max Welling in 2017.[9]

A GCN layer defines a first-order approximation of a localized spectral filter on graphs. GCNs can be understood as a generalization of convolutional neural networks to graph-structured data.

The formal expression of a GCN layer reads as follows:

$$\mathbf{H} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{\Theta}\right)$$

where $\mathbf{H}$ is the matrix of node representations $\mathbf{h}_u$, $\mathbf{X}$ is the matrix of node features $\mathbf{x}_u$, $\sigma(\cdot)$ is an activation function (e.g., ReLU), $\tilde{\mathbf{A}}$ is the graph adjacency matrix with the addition of self-loops, $\tilde{\mathbf{D}}$ is the graph degree matrix with the addition of self-loops, and $\mathbf{\Theta}$ is a matrix of trainable parameters.

In particular, let $\mathbf{A}$ be the graph adjacency matrix: then, one can define $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}_{ii} = \sum_{j \in V} \tilde{A}_{ij}$, where $\mathbf{I}$ denotes the identity matrix. This normalization ensures that the eigenvalues of $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ are bounded in the range $[0, 1]$, avoiding numerical instabilities and exploding/vanishing gradients.

A limitation of GCNs is that they do not allow multidimensional edge features $\mathbf{e}_{uv}$.[9] It is however possible to associate scalar weights w_{uv} to each edge by imposing $A_{uv} = w_{uv}$, i.e., by setting each nonzero entry in the adjacency matrix equal to the weight of the corresponding edge.

Graph attention network

The graph attention network (GAT) was introduced by Petar Veličković et al. in 2018.[11]

Graph attention network is a combination of a GNN and an attention layer. The implementation of attention layer in graphical neural networks helps provide attention or focus to the important information from the data instead of focusing on the whole data.

A multi-head GAT layer can be expressed as follows:

$$\mathbf{h}_u = \overset{K}{\underset{k=1}{\Big\|}} \sigma\left(\sum_{v \in N_u} \alpha_{uv} \mathbf{W}^k \mathbf{x}_v\right)$$

where K is the number of attention heads, $\Big\|$ denotes vector concatenation, $\sigma(\cdot)$ is an activation function (e.g., ReLU), α_{ij} are attention coefficients, and W^k is a matrix of trainable parameters for the k-th attention head.

For the final GAT layer, the outputs from each attention head are averaged before the application of the activation function. Formally, the final GAT layer can be written as:

$$\mathbf{h}_u = \sigma\left(\frac{1}{K} \sum_{k=1}^{K} \sum_{v \in N_u} \alpha_{uv} \mathbf{W}^k \mathbf{x}_v\right)$$

Attention in Machine Learning is a technique that mimics cognitive attention. In the context of learning on graphs, the attention coefficient α_{uv} measures *how important* is node $u \in V$ to node $v \in V$.

Normalized attention coefficients are computed as follows:

$$\alpha_{uv} = \frac{\exp(\text{LeakyReLU}\left(\mathbf{a}^T [\mathbf{W}\mathbf{x}_u \| \mathbf{W}\mathbf{x}_v \| \mathbf{e}_{uv}]\right))}{\sum_{z \in N_u} \exp(\text{LeakyReLU}\left(\mathbf{a}^T [\mathbf{W}\mathbf{x}_u \| \mathbf{W}\mathbf{x}_z \| \mathbf{e}_{uz}]\right))}$$

where $\mathbf{a}$ is a vector of learnable weights, $\cdot^T$ indicates transposition, $\mathbf{e}_{uv}$ are the edge features (if present), and LeakyReLU is a modified ReLU activation function. Attention coefficients are normalized to make them easily comparable across different nodes.[11]

A GCN can be seen as a special case of a GAT where attention coefficients are not learnable, but fixed and equal to the edge weights w_{uv}.

Gated graph sequence neural network

The gated graph sequence neural network (GGS-NN) was introduced by Yujia Li et al. in 2015.[39] The GGS-NN extends the GNN formulation by Scarselli et al.[2] to output sequences. The message passing framework is implemented as an update rule to a gated recurrent unit (GRU) cell.

A GGS-NN can be expressed as follows:

$$\mathbf{h}_u^{(0)} = \mathbf{x}_u \parallel \mathbf{0}$$
$$\mathbf{m}_u^{(l+1)} = \sum_{v \in N_u} \mathbf{\Theta} \mathbf{h}_v$$
$$\mathbf{h}_u^{(l+1)} = \text{GRU}(\mathbf{m}_u^{(l+1)}, \mathbf{h}_u^{(l)})$$

where $\parallel$ denotes vector concatenation, $\mathbf{0}$ is a vector of zeros, $\mathbf{\Theta}$ is a matrix of learnable parameters, GRU is a GRU cell, and l denotes the sequence index. In a GGS-NN, the node representations are regarded as the hidden states of a GRU cell. The initial node features $\mathbf{x}_u^{(0)}$ are zero-padded up to the hidden state dimension of the GRU cell. The same GRU cell is used for updating representations for each node.

Local pooling layers

Local pooling layers coarsen the graph via downsampling. Subsequently, several learnable local pooling strategies that have been proposed are presented.[31] For each case, the input is the initial graph represented by a matrix $\mathbf{X}$ of node features, and the graph adjacency matrix $\mathbf{A}$. The output is the new matrix $\mathbf{X}'$ of node features, and the new graph adjacency matrix $\mathbf{A}'$.

Top-k pooling

We first set

$$\mathbf{y} = \frac{\mathbf{X}\mathbf{p}}{\lVert \mathbf{p} \rVert}$$

where $\mathbf{p}$ is a learnable projection vector. The projection vector $\mathbf{p}$ computes a scalar projection value for each graph node.

The top-k pooling layer [29] can then be formalised as follows:

$$\mathbf{X}' = (\mathbf{X} \odot \text{sigmoid}(\mathbf{y}))_{\mathbf{i}}$$

$$\mathbf{A}' = \mathbf{A}_{\mathbf{i},\mathbf{i}}$$

where $\mathbf{i} = \text{top}_k(\mathbf{y})$ is the subset of nodes with the top-k highest projection scores, $\odot$ denotes element-wise matrix multiplication, and $\text{sigmoid}(\cdot)$ is the sigmoid function. In other words, the nodes with the top-k highest projection scores are retained in the new adjacency matrix $\mathbf{A}'$. The $\text{sigmoid}(\cdot)$ operation makes the projection vector $\mathbf{p}$ trainable by backpropagation, which otherwise would produce discrete outputs.[29]

Self-attention pooling

We first set

$$\mathbf{y} = \text{GNN}(\mathbf{X}, \mathbf{A})$$

where GNN is a generic permutation equivariant GNN layer (e.g., GCN, GAT, MPNN).

The Self-attention pooling layer[30] can then be formalised as follows:

$$\mathbf{X}' = (\mathbf{X} \odot \mathbf{y})_{\mathbf{i}}$$

$$\mathbf{A}' = \mathbf{A}_{\mathbf{i},\mathbf{i}}$$

where $\mathbf{i} = \text{top}_k(\mathbf{y})$ is the subset of nodes with the top-k highest projection scores, $\odot$ denotes element-wise matrix multiplication.

The self-attention pooling layer can be seen as an extension of the top-k pooling layer. Differently from top-k pooling, the self-attention scores computed in self-attention pooling account both for the graph features and the graph topology.

Heterophilic Graph Learning

Homophily principle, i.e., nodes with the same labels or similar attributes are more likely to be connected, has been commonly believed to be the main reason for the superiority of Graph Neural Networks (GNNs) over traditional Neural Networks (NNs) on graph-structured data, especially on node-level tasks.[41] However, recent work has identified a non-trivial set of datasets where GNN's performance compared to the NN's is not satisfactory.[42] Heterophily, i.e., low homophily, has been considered the main cause of this empirical observation.[43] People have begun to revisit and re-evaluate most existing graph models in the heterophily scenario across various kinds of graphs, e.g., heterogeneous graphs, temporal graphs and hypergraphs. Moreover, numerous graph-related applications are found to be closely related to the heterophily problem, e.g. graph fraud/anomaly detection, graph adversarial attacks and robustness, privacy, federated learning and point cloud segmentation, graph clustering, recommender systems, generative models, link prediction, graph classification and coloring, etc. In the past few years, considerable effort has been devoted to studying and addressing the heterophily issue in graph learning.[41][43][44]

Applications

Protein folding

Graph neural networks are one of the main building blocks of AlphaFold, an artificial intelligence program developed by Google's DeepMind for solving the protein folding problem in biology. AlphaFold achieved first place in several CASP competitions.[45][46][40]

Social networks

Social networks are a major application domain for GNNs due to their natural representation as social graphs. GNNs are used to develop recommender systems based on both social relations and item relations.[47][16]

Combinatorial optimization

GNNs are used as fundamental building blocks for several combinatorial optimization algorithms.[48] Examples include computing shortest paths or Eulerian circuits for a given graph,[39] deriving chip placements superior or competitive to handcrafted human solutions,[49] and improving expert-designed branching rules in branch and bound.[50]

Cyber security

When viewed as a graph, a network of computers can be analyzed with GNNs for anomaly detection. Anomalies within provenance graphs often correlate to malicious activity within the network. GNNs have been used to identify these anomalies on individual nodes[51] and within paths[52] to detect malicious processes, or on the edge level[53] to detect lateral movement.

Water distribution networks

Water distribution systems can be modelled as graphs, being then a straightforward application of GNN. This kind of algorithm has been applied to water demand forecasting,[54] interconnecting District Measuring Areas to improve the forecasting capacity. Other application of this algorithm on water distribution modelling is the development of metamodels.[55]

Computer Vision

To represent an image as a graph structure, the image is first divided into multiple patches, each of which is treated as a node in the graph. Edges are then formed by connecting each node to its nearest neighbors based on spatial or feature similarity. This graph-based representation enables the application of graph learning models to visual tasks. The relational structure helps to enhance feature extraction and improve performance on image understanding.[56]

Text and NLP

Graph-based representation of text helps to capture deeper semantic relationships between words. Many studies have used graph networks to enhance performance in various text processing tasks such as text classification, question answering, Neural Machine Translation (NMT), event extraction, fact verification, etc.[57]

References

1. Wu, Lingfei; Cui, Peng; Pei, Jian; Zhao, Liang (2022). "Graph Neural Networks: Foundations, Frontiers, and Applications" (https://graph-neural-networks.github.io/). *Springer Singapore*: 725.
2. Scarselli, Franco; Gori, Marco; Tsoi, Ah Chung; Hagenbuchner, Markus; Monfardini, Gabriele (2009). "The Graph Neural Network Model". *IEEE Transactions on Neural Networks*. **20** (1): 61–80. doi:10.1109/TNN.2008.2005605 (https://doi.org/10.1109%2FTNN.2008.2005605). ISSN 1941-0093 (https://search.worldcat.org/issn/1941-0093). PMID 19068426 (https://pubmed.ncbi.nlm.nih.gov/19068426). S2CID 206756462 (https://api.semanticscholar.org/CorpusID:206756462).

3. Micheli, Alessio (2009). "Neural Network for Graphs: A Contextual Constructive Approach". *IEEE Transactions on Neural Networks*. **20** (3): 498–511. doi:10.1109/TNN.2008.2010350 (https://doi.org/10.1109%2FTNN.2008.2010350). ISSN 1045-9227 (https://search.worldcat.org/issn/1045-9227). PMID 19193509 (https://pubmed.ncbi.nlm.nih.gov/19193509). S2CID 17486263 (https://api.semanticscholar.org/CorpusID:17486263).
4. Sanchez-Lengeling, Benjamin; Reif, Emily; Pearce, Adam; Wiltschko, Alex (2 September 2021). "A Gentle Introduction to Graph Neural Networks" (https://distill.pub/2021/gnn-intro). *Distill*. **6** (9): e33. doi:10.23915/distill.00033 (https://doi.org/10.23915%2Fdistill.00033). ISSN 2476-0757 (https://search.worldcat.org/issn/2476-0757).
5. Daigavane, Ameya; Ravindran, Balaraman; Aggarwal, Gaurav (2 September 2021). "Understanding Convolutions on Graphs" (https://distill.pub/2021/understanding-gnns). *Distill*. **6** (9): e32. doi:10.23915/distill.00032 (https://doi.org/10.23915%2Fdistill.00032). ISSN 2476-0757 (https://search.worldcat.org/issn/2476-0757). S2CID 239678898 (https://api.semanticscholar.org/CorpusID:239678898).
6. Stokes, Jonathan M.; Yang, Kevin; Swanson, Kyle; Jin, Wengong; Cubillos-Ruiz, Andres; Donghia, Nina M.; MacNair, Craig R.; French, Shawn; Carfrae, Lindsey A.; Bloom-Ackermann, Zohar; Tran, Victoria M.; Chiappino-Pepe, Anush; Badran, Ahmed H.; Andrews, Ian W.; Chory, Emma J. (20 February 2020). "A Deep Learning Approach to Antibiotic Discovery" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8349178). *Cell*. **180** (4): 688–702.e13. doi:10.1016/j.cell.2020.01.021 (https://doi.org/10.1016%2Fj.cell.2020.01.021). ISSN 1097-4172 (https://search.worldcat.org/issn/1097-4172). PMC 8349178 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8349178). PMID 32084340 (https://pubmed.ncbi.nlm.nih.gov/32084340).
7. Yang, Kevin; Swanson, Kyle; Jin, Wengong; Coley, Connor; Eiden, Philipp; Gao, Hua; Guzman-Perez, Angel; Hopper, Timothy; Kelley, Brian (20 November 2019). "Analyzing Learned Molecular Representations for Property Prediction". arXiv:1904.01561 (https://arxiv.org/abs/1904.01561) [cs.LG (https://arxiv.org/archive/cs.LG)].
8. Marchant, Jo (20 February 2020). "Powerful antibiotics discovered using AI" (https://www.nature.com/articles/d41586-020-00018-3). *Nature*. doi:10.1038/d41586-020-00018-3 (https://doi.org/10.1038%2Fd41586-020-00018-3). PMID 33603175 (https://pubmed.ncbi.nlm.nih.gov/33603175).
9. Kipf, Thomas N; Welling, Max (2016). "Semi-supervised classification with graph convolutional networks". *IEEE Transactions on Neural Networks*. **5** (1): 61–80. arXiv:1609.02907 (https://arxiv.org/abs/1609.02907). doi:10.1109/TNN.2008.2005605 (https://doi.org/10.1109%2FTNN.2008.2005605). PMID 19068426 (https://pubmed.ncbi.nlm.nih.gov/19068426). S2CID 206756462 (https://api.semanticscholar.org/CorpusID:206756462).
10. Hamilton, William; Ying, Rex; Leskovec, Jure (2017). "Inductive Representation Learning on Large Graphs" (https://cs.stanford.edu/people/jure/pubs/graphsage-nips17.pdf) (PDF). *Neural Information Processing Systems*. **31**. arXiv:1706.02216 (https://arxiv.org/abs/1706.02216) – via Stanford.
11. Veličković, Petar; Cucurull, Guillem; Casanova, Arantxa; Romero, Adriana; Liò, Pietro; Bengio, Yoshua (4 February 2018). "Graph Attention Networks". arXiv:1710.10903 (https://arxiv.org/abs/1710.10903) [stat.ML (https://arxiv.org/archive/stat.ML)].
12. Bronstein, Michael M.; Bruna, Joan; Cohen, Taco; Veličković, Petar (4 May 2021). "Geometric Deep Learning: Grids, Groups, Graphs Geodesics and Gauges". arXiv:2104.13478 (https://arxiv.org/abs/2104.13478) [cs.LG (https://arxiv.org/archive/cs.LG)].
13. Hajij, M.; Zamzmi, G.; Papamarkou, T.; Miolane, N.; Guzmán-Sáenz, A.; Ramamurthy, K. N.; Schaub, M. T. (2022). "Topological deep learning: Going beyond graph data". arXiv:2206.00606 (https://arxiv.org/abs/2206.00606) [cs.LG (https://arxiv.org/archive/cs.LG)].
14. Veličković, Petar (2022). "Message passing all the way up". arXiv:2202.11097 (https://arxiv.org/abs/2202.11097) [cs.LG (https://arxiv.org/archive/cs.LG)].

15. Wu, Lingfei; Chen, Yu; Shen, Kai; Guo, Xiaojie; Gao, Hanning; Li, Shucheng; Pei, Jian; Long, Bo (2023). "Graph Neural Networks for Natural Language Processing: A Survey" (https://www.nowpublishers.com/article/Details/MAL-096). *Foundations and Trends in Machine Learning*. **16** (2): 119–328. arXiv:2106.06090 (https://arxiv.org/abs/2106.06090). doi:10.1561/2200000096 (https://doi.org/10.1561%2F2200000096). ISSN 1941-0093 (https://search.worldcat.org/issn/1941-0093). PMID 19068426 (https://pubmed.ncbi.nlm.nih.gov/19068426). S2CID 206756462 (https://api.semanticscholar.org/CorpusID:206756462).
16. Ying, Rex; He, Ruining; Chen, Kaifeng; Eksombatchai, Pong; Hamilton, William L.; Leskovec, Jure (2018). *Graph Convolutional Neural Networks for Web-Scale Recommender Systems*. pp. 974–983. arXiv:1806.01973 (https://arxiv.org/abs/1806.01973). doi:10.1145/3219819.3219890 (https://doi.org/10.1145%2F3219819.3219890). ISBN 9781450355520. S2CID 46949657 (https://api.semanticscholar.org/CorpusID:46949657).
17. "Stanford Large Network Dataset Collection" (https://snap.stanford.edu/data/). *snap.stanford.edu*. Retrieved 5 July 2021.
18. Zhang, Weihang; Cui, Yang; Liu, Bowen; Loza, Martin; Park, Sung-Joon; Nakai, Kenta (5 April 2024). "HyGAnno: Hybrid graph neural network-based cell type annotation for single-cell ATAC sequencing data" (https://academic.oup.com/bib/article/25/3/bbae152/7641197). *Briefings in Bioinformatics*. **25** (3): bbae152. doi:10.1093/bib/bbae152 (https://doi.org/10.1093%2Fbib%2Fbbae152). PMC 10998639 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10998639). PMID 38581422 (https://pubmed.ncbi.nlm.nih.gov/38581422).
19. Gilmer, Justin; Schoenholz, Samuel S.; Riley, Patrick F.; Vinyals, Oriol; Dahl, George E. (17 July 2017). "Neural Message Passing for Quantum Chemistry" (http://proceedings.mlr.press/v70/gilmer17a.html). *Proceedings of Machine Learning Research*: 1263–1272. arXiv:1704.01212 (https://arxiv.org/abs/1704.01212).
20. Coley, Connor W.; Jin, Wengong; Rogers, Luke; Jamison, Timothy F.; Jaakkola, Tommi S.; Green, William H.; Barzilay, Regina; Jensen, Klavs F. (2 January 2019). "A graph-convolutional neural network model for the prediction of chemical reactivity" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6335848). *Chemical Science*. **10** (2): 370–377. doi:10.1039/C8SC04228D (https://doi.org/10.1039%2FC8SC04228D). ISSN 2041-6539 (https://search.worldcat.org/issn/2041-6539). PMC 6335848 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6335848). PMID 30746086 (https://pubmed.ncbi.nlm.nih.gov/30746086).
21. Qasim, Shah Rukh; Kieseler, Jan; Iiyama, Yutaro; Pierini, Maurizio Pierini (2019). "Learning representations of irregular particle-detector geometry with distance-weighted graph networks" (https://doi.org/10.1140%2Fepjc%2Fs10052-019-7113-9). *The European Physical Journal C*. **79** (7): 608. arXiv:1902.07987 (https://arxiv.org/abs/1902.07987). Bibcode:2019EPJC...79..608Q (https://ui.adsabs.harvard.edu/abs/2019EPJC...79..608Q). doi:10.1140/epjc/s10052-019-7113-9 (https://doi.org/10.1140%2Fepjc%2Fs10052-019-7113-9). S2CID 88518244 (https://api.semanticscholar.org/CorpusID:88518244).
22. Li, Zhuwen; Chen, Qifeng; Koltun, Vladlen (2018). "Text Simplification with Self-Attention-Based Pointer-Generator Networks". *Neural Information Processing*. Lecture Notes in Computer Science. Vol. 31. pp. 537–546. arXiv:1810.10659 (https://arxiv.org/abs/1810.10659). doi:10.1007/978-3-030-04221-9_48 (https://doi.org/10.1007%2F978-3-030-04221-9_48). ISBN 978-3-030-04220-2.
23. Matthias, Fey; Lenssen, Jan E. (2019). "Fast Graph Representation Learning with PyTorch Geometric". arXiv:1903.02428 (https://arxiv.org/abs/1903.02428) [cs.LG (https://arxiv.org/archive/cs.LG)].
24. "Tensorflow GNN" (https://github.com/tensorflow/gnn). *GitHub*. Retrieved 30 June 2022.
25. "Deep Graph Library (DGL)" (https://www.dgl.ai/). Retrieved 12 September 2024.
26. "jraph" (https://github.com/deepmind/jraph). *GitHub*. Retrieved 30 June 2022.
27. Lucibello, Carlo (2021). "GraphNeuralNetworks.jl" (https://github.com/CarloLucibello/GraphNeuralNetworks.jl). *GitHub*. Retrieved 21 September 2023.
28. *FluxML/GeometricFlux.jl* (https://github.com/FluxML/GeometricFlux.jl), FluxML, 31 January 2024, retrieved 3 February 2024

29. Gao, Hongyang; Ji, Shuiwang Ji (2019). "Graph U-Nets". arXiv:1905.05178 (https://arxiv.org/abs/1905.05178) [cs.LG (https://arxiv.org/archive/cs.LG)].
30. Lee, Junhyun; Lee, Inyeop; Kang, Jaewoo (2019). "Self-Attention Graph Pooling". arXiv:1904.08082 (https://arxiv.org/abs/1904.08082) [cs.LG (https://arxiv.org/archive/cs.LG)].
31. Liu, Chuang; Zhan, Yibing; Li, Chang; Du, Bo; Wu, Jia; Hu, Wenbin; Liu, Tongliang; Tao, Dacheng (2022). "Graph Pooling for Graph Neural Networks: Progress, Challenges, and Opportunities". arXiv:2204.07321 (https://arxiv.org/abs/2204.07321) [cs.LG (https://arxiv.org/archive/cs.LG)].
32. Douglas, B. L. (27 January 2011). "The Weisfeiler–Lehman Method and Graph Isomorphism Testing". arXiv:1101.5211 (https://arxiv.org/abs/1101.5211) [math.CO (https://arxiv.org/archive/math.CO)].
33. Xu, Keyulu; Hu, Weihua; Leskovec, Jure; Jegelka, Stefanie (22 February 2019). "How Powerful are Graph Neural Networks?". arXiv:1810.00826 (https://arxiv.org/abs/1810.00826) [cs.LG (https://arxiv.org/archive/cs.LG)].
34. Bodnar, Christian; Frasca, Fabrizio; Guang Wang, Yu; Otter, Nina; Montúfar, Guido; Liò, Pietro; Bronstein, Michael (2021). "Weisfeiler and Lehman Go Topological: Message Passing Simplicial Networks". arXiv:2103.03212 (https://arxiv.org/abs/2103.03212) [cs.LG (https://arxiv.org/archive/cs.LG)].
35. Grady, Leo; Polimeni, Jonathan (2011). *Discrete Calculus: Applied Analysis on Graphs for Computational Science* (http://leogrady.net/wp-content/uploads/2017/01/grady2010discrete.pdf) (PDF). Springer.
36. Chen, Deli; Lin, Yankai; Li, Wei; Li, Peng; Zhou, Jie; Sun, Xu (2020). "Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View". *Proceedings of the AAAI Conference on Artificial Intelligence*. **34** (4): 3438–3445. arXiv:1909.03211 (https://arxiv.org/abs/1909.03211). doi:10.1609/aaai.v34i04.5747 (https://doi.org/10.1609%2Faaai.v34i04.5747). S2CID 202539008 (https://api.semanticscholar.org/CorpusID:202539008).
37. Alon, Uri; Yahav, Eran (2021). "On the Bottleneck of Graph Neural Networks and its Practical Implications". arXiv:2006.05205 (https://arxiv.org/abs/2006.05205) [cs.LG (https://arxiv.org/archive/cs.LG)].
38. Xu, Keyulu; Zhang, Mozhi; Jegelka, Stephanie; Kawaguchi, Kenji (2021). "Optimization of Graph Neural Networks: Implicit Acceleration by Skip Connections and More Depth". arXiv:2105.04550 (https://arxiv.org/abs/2105.04550) [cs.LG (https://arxiv.org/archive/cs.LG)].
39. Li, Yujia; Tarlow, Daniel; Brockschmidt, Mark; Zemel, Richard (2016). "Gated Graph Sequence Neural Networks". arXiv:1511.05493 (https://arxiv.org/abs/1511.05493) [cs.LG (https://arxiv.org/archive/cs.LG)].
40. Xu, Keyulu; Li, Chengtao; Tian, Yonglong; Sonobe, Tomohiro; Kawarabayashi, Ken-ichi; Jegelka, Stefanie (2018). "Representation Learning on Graphs with Jumping Knowledge Networks". arXiv:1806.03536 (https://arxiv.org/abs/1806.03536) [cs.LG (https://arxiv.org/archive/cs.LG)].
41. Luan, Sitao; Hua, Chenqing; Lu, Qincheng; Ma, Liheng; Wu, Lirong; Wang, Xinyu; Xu, Minkai; Chang, Xiao-Wen; Precup, Doina; Ying, Rex; Li, Stan Z.; Tang, Jian; Wolf, Guy; Jegelka, Stefanie (2024). "The Heterophilic Graph Learning Handbook: Benchmarks, Models, Theoretical Analysis, Applications and Challenges". arXiv:2407.09618 (https://arxiv.org/abs/2407.09618) [cs.LG (https://arxiv.org/archive/cs.LG)].
42. Luan, Sitao; Hua, Chenqing; Lu, Qincheng; Zhu, Jiaqi; Chang, Xiao-Wen; Precup, Doina (2024). "When do We Need Graph Neural Networks for Node Classification?" (https://link.springer.com/chapter/10.1007/978-3-031-53468-3_4). In Cherifi, Hocine; Rocha, Luis M.; Cherifi, Chantal; Donduran, Murat (eds.). *Complex Networks & Their Applications XII*. Studies in Computational Intelligence. Vol. 1141. Cham: Springer Nature Switzerland. pp. 37–48. doi:10.1007/978-3-031-53468-3_4 (https://doi.org/10.1007%2F978-3-031-53468-3_4). ISBN 978-3-031-53467-6.

43. Luan, Sitao; Hua, Chenqing; Lu, Qincheng; Zhu, Jiaqi; Zhao, Mingde; Zhang, Shuyuan; Chang, Xiao-Wen; Precup, Doina (6 December 2022). "Revisiting Heterophily For Graph Neural Networks" (https://proceedings.neurips.cc/paper_files/paper/2022/hash/092359ce5cf60a80e882378944bf1be4-Abstract-Conference.html). *Advances in Neural Information Processing Systems*. **35**: 1362–1375. arXiv:2210.07606 (https://arxiv.org/abs/2210.07606).
44. Luan, Sitao; Hua, Chenqing; Xu, Minkai; Lu, Qincheng; Zhu, Jiaqi; Chang, Xiao-Wen; Fu, Jie; Leskovec, Jure; Precup, Doina (15 December 2023). "When Do Graph Neural Networks Help with Node Classification? Investigating the Homophily Principle on Node Distinguishability" (https://proceedings.neurips.cc/paper_files/paper/2023/hash/5ba11de4c74548071899cf41dec078bf-Abstract-Conference.html). *Advances in Neural Information Processing Systems*. **36**: 28748–28760.
45. Sample, Ian (2 December 2018). "Google's DeepMind predicts 3D shapes of proteins" (https://www.theguardian.com/science/2018/dec/02/google-deepminds-ai-program-alphafold-predicts-3d-shapes-of-proteins). *The Guardian*. Retrieved 30 November 2020.
46. "DeepMind's protein-folding AI has solved a 50-year-old grand challenge of biology" (https://www.technologyreview.com/2020/11/30/1012712/deepmind-protein-folding-ai-solved-biology-science-drugs-disease/). *MIT Technology Review*. Retrieved 30 November 2020.
47. Fan, Wenqi; Ma, Yao; Li, Qing; He, Yuan; Zhao, Eric; Tang, Jiliang; Yin, Dawei (2019). *Graph Neural Networks for Social Recommendation*. pp. 417–426. arXiv:1902.07243 (https://arxiv.org/abs/1902.07243). doi:10.1145/3308558.3313488 (https://doi.org/10.1145%2F3308558.3313488). hdl:10397/81232 (https://hdl.handle.net/10397%2F81232). ISBN 9781450366748. S2CID 67769538 (https://api.semanticscholar.org/CorpusID:67769538).
48. Cappart, Quentin; Chételat, Didier; Khalil, Elias; Lodi, Andrea; Morris, Christopher; Veličković, Petar (2021). "Combinatorial optimization and reasoning with graph neural networks". arXiv:2102.09544 (https://arxiv.org/abs/2102.09544) [cs.LG (https://arxiv.org/archive/cs.LG)].
49. Mirhoseini, Azalia; Goldie, Anna; Yazgan, Mustafa; Jiang, Joe Wenjie; Songhori, Ebrahim; Wang, Shen; Lee, Young-Joon; Johnson, Eric; Pathak, Omkar; Nazi, Azade; Pak, Jiwoo; Tong, Andy; Srinivasa, Kavya; Hang, William; Tuncer, Emre; Le, Quoc V.; Laudon, James; Ho, Richard; Carpenter, Roger; Dean, Jeff (2021). "A graph placement methodology for fast chip design". *Nature*. **594** (7862): 207–212. Bibcode:2021Natur.594..207M (https://ui.adsabs.harvard.edu/abs/2021Natur.594..207M). doi:10.1038/s41586-021-03544-w (https://doi.org/10.1038%2Fs41586-021-03544-w). PMID 34108699 (https://pubmed.ncbi.nlm.nih.gov/34108699). S2CID 235395490 (https://api.semanticscholar.org/CorpusID:235395490).
50. Gasse, Maxime; Chételat, Didier; Ferroni, Nicola; Charlin, Laurent; Lodi, Andrea (2019). "Exact Combinatorial Optimization with Graph Convolutional Neural Networks". arXiv:1906.01629 (https://arxiv.org/abs/1906.01629) [cs.LG (https://arxiv.org/archive/cs.LG)].
51. Wang, Su; Wang, Zhiliang; Zhou, Tao; Sun, Hongbin; Yin, Xia; Han, Dongqi; Zhang, Han; Shi, Xingang; Yang, Jiahai (2022). "Threatrace: Detecting and Tracing Host-Based Threats in Node Level Through Provenance Graph Learning" (https://ieeexplore.ieee.org/document/9899459/;jsessionid=NzAXdLahhjEX-xmrFzOROk4qxoaz40aJFvKcZRgjck8-zCOucJi7!380715771). *IEEE Transactions on Information Forensics and Security*. **17**: 3972–3987. arXiv:2111.04333 (https://arxiv.org/abs/2111.04333). doi:10.1109/TIFS.2022.3208815 (https://doi.org/10.1109%2FTIFS.2022.3208815). ISSN 1556-6021 (https://search.worldcat.org/issn/1556-6021). S2CID 243847506 (https://api.semanticscholar.org/CorpusID:243847506).
52. Wang, Qi; Hassan, Wajih Ul; Li, Ding; Jee, Kangkook; Yu, Xiao (2020). "You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis" (https://doi.org/10.14722%2Fndss.2020.24167). *Network and Distributed Systems Security Symposium*. doi:10.14722/ndss.2020.24167 (https://doi.org/10.14722%2Fndss.2020.24167). ISBN 978-1-891562-61-7. S2CID 211267791 (https://api.semanticscholar.org/CorpusID:211267791).
53. King, Isaiah J.; Huang, H. Howie (2022). "Euler: Detecting Network Lateral Movement via Scalable Temporal Link Prediction" (https://www.ndss-symposium.org/wp-content/uploads/2022-107A-paper.pdf) (PDF). *In Proceedings of the 29th Network and Distributed Systems Security Symposium*. doi:10.14722/ndss.2022.24107 (https://doi.org/10.14722%2Fndss.2022.24107). S2CID 248221601 (https://api.semanticscholar.org/CorpusID:248221601).

54. Zanfei, Ariele; et al. (2022). "Graph Convolutional Recurrent Neural Networks for Water Demand Forecasting" (https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2022WR032299). *Water Resources Research*. **58** (7). AGU. Bibcode:2022WRR....5832299Z (https://ui.adsabs.harvard.edu/abs/2022WRR....5832299Z). doi:10.1029/2022WR032299 (https://doi.org/10.1029%2F2022WR032299). Retrieved 11 June 2024.
55. Zanfei, Ariele; et al. (2023). "Shall we always use hydraulic models? A graph neural network metamodel for water system calibration and uncertainty assessment" (https://www.sciencedirect.com/science/article/abs/pii/S0043135423007005). *Water Research*. **242** 120264. Bibcode:2023WatRe.24220264Z (https://ui.adsabs.harvard.edu/abs/2023WatRe.24220264Z). doi:10.1016/j.watres.2023.120264 (https://doi.org/10.1016%2Fj.watres.2023.120264). PMID 37393807 (https://pubmed.ncbi.nlm.nih.gov/37393807). Retrieved 11 June 2024.
56. Han, Kai; Wang, Yunhe; Guo, Jianyuan; Tang, Yehui; Wu, Enhua (2022). "Vision GNN: An Image is Worth Graph of Nodes". arXiv:2206.00272 (https://arxiv.org/abs/2206.00272) [cs.CV (https://arxiv.org/archive/cs.CV)].
57. Zhou, Jie; Cui, Ganqu; Hu, Shengding; Zhang, Zhengyan; Yang, Cheng; Liu, Zhiyuan; Wang, Lifeng; Li, Changcheng; Sun, Maosong (1 January 2020). "Graph neural networks: A review of methods and applications" (https://doi.org/10.1016%2Fj.aiopen.2021.01.001). *AI Open*. **1**: 57–81. doi:10.1016/j.aiopen.2021.01.001 (https://doi.org/10.1016%2Fj.aiopen.2021.01.001). ISSN 2666-6510 (https://search.worldcat.org/issn/2666-6510).

External links

- A Gentle Introduction to Graph Neural Networks (https://distill.pub/2021/gnn-intro/)