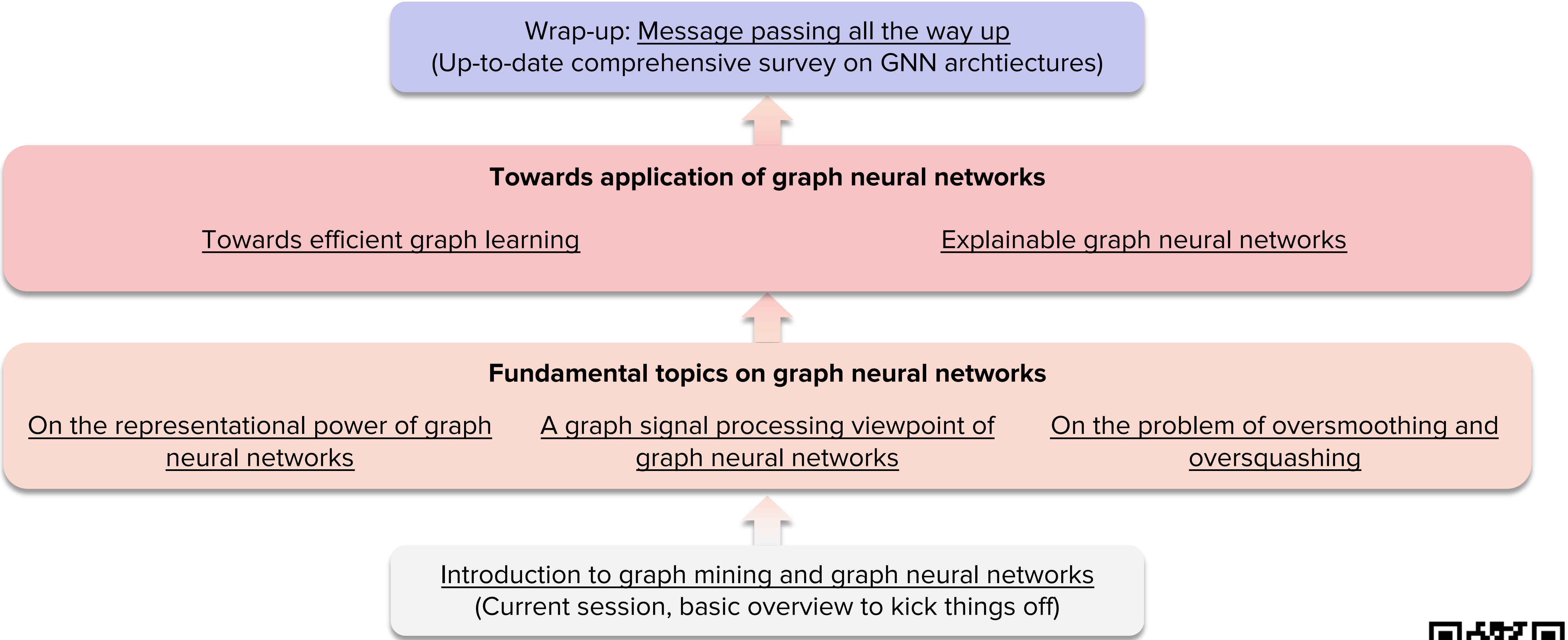
Seminar Series on Graph Neural Networks 01

Introduction to Graph Mining and Graph Neural Networks

Yong-Min Shin
School of Mathematics and Computing (Computational Science and Engineering)
Yonsei University

2025.03.31

Before going in....



Wrap-up: Message passing all the way up
(Up-to-date comprehensive survey on GNN archtiectures)

Towards application of graph neural networks

Towards efficient graph learning

Explainable graph neural networks

Fundamental topics on graph neural networks

On the representational power of graph neural networks

A graph signal processing viewpoint of graph neural networks

On the problem of oversmoothing and oversquashing

Introduction to graph mining and graph neural networks
(Current session, basic overview to kick things off)

(Some of the topics may change in the future for a better alternative)

* Presentation slides are available at:
(jordan7186.github.io/presentations/)

Objectives

1. Understanding of **graphs** as a **general data type**
2. Understanding of the general framework of **graph neural networks (GNNs)**
3. High-level understanding of **several key GNN architectures: GCN & GraphSAGE**

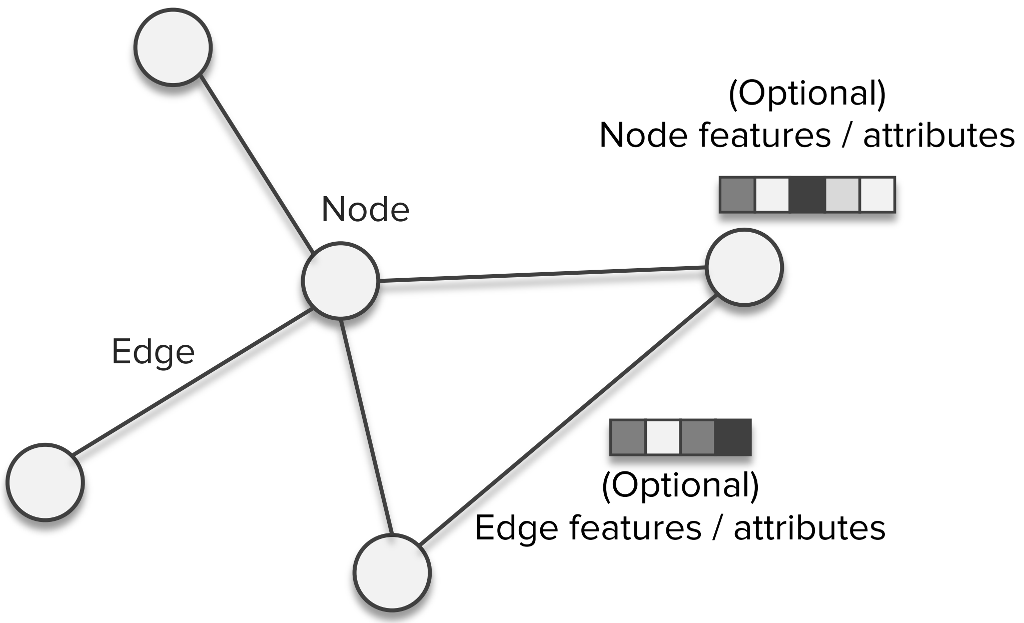
Understanding of graphs as a general data type

*This part is heavily influenced by one of my academic heros, Petar Veličković. These are some materials from his public materials that I have referred to:

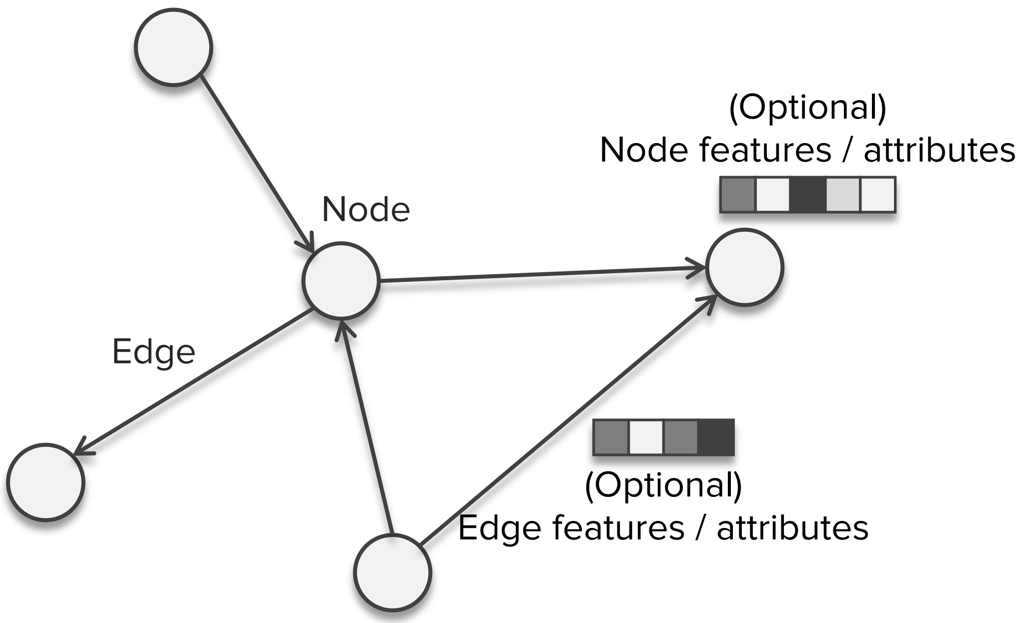
- (Slide) Everything is Connected: Graph Neural Networks from the Ground Up (2021)
- (Blog) Graph & Geometric ML in 2024: Where We Are and What's Next (Part II – Applications)

Graphs as an abstract datatype

Graphs are an abstract type of data where nodes (entities) are **connected** by edges (connections)



Undirected graph



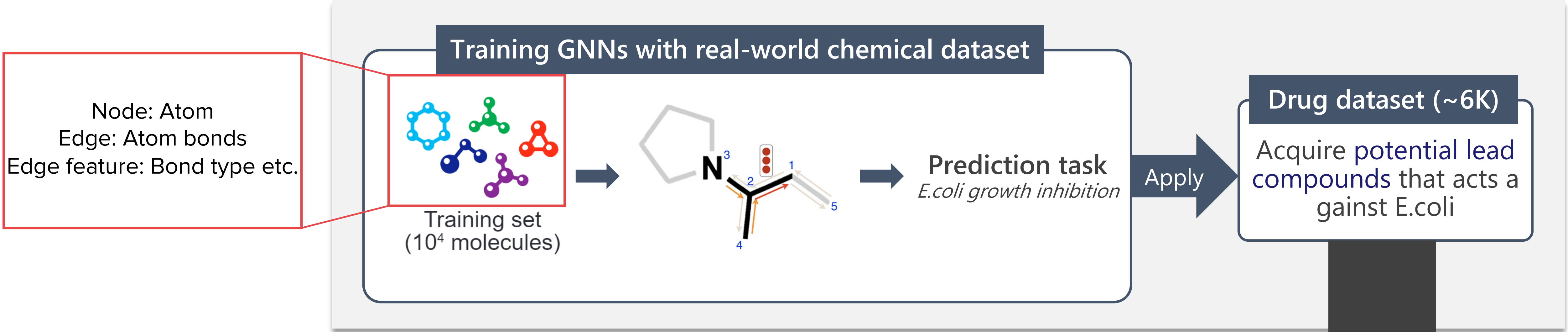
Directed graph

...But honestly, looking at this does not result in a **practical** understanding of graphs.

Therefore, we will look at **various applications** in the field of **graph machine learning** before moving our discussion further.

Area 1) Biology & Chemistry Research

Example 1: The discovery of Halicin, GNN-guided antibiotic discovery



Node: Atom
Edge: Atom bonds
Edge feature: Bond type etc.

Training GNNs with real-world chemical dataset

Prediction task
E.coli growth inhibition

Apply

Drug dataset (~6K)

Acquire potential lead compounds that acts a gainst E.coli

Further screening
& Empirical testing

Cell

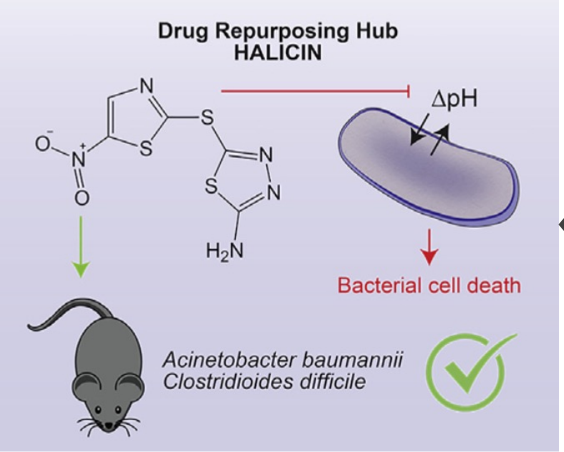
Article

A Deep Learning Approach to Antibiotic Discovery

Jonathan M. Stokes,[1,2,3] Kevin Yang,[3,4,10] Kyle Swanson,[3,4,10] Wengong Jin,[3,4] Andres Cubillos-Ruiz,[1,2,5] Nina M. Donghia,[1,5] Craig R. MacNair,[6] Shawn French,[6] Lindsey A. Carfrae,[6] Zohar Bloom-Ackermann,[2,7] Victoria M. Tran,[2] Anush Chiappino-Pepe,[5,7] Ahmed H. Badran,[2] Ian W. Andrews,[1,2,5] Emma J. Chory,[1,2] George M. Church,[5,7,8] Eric D. Brown,[6] Tommi S. Jaakkola,[3,4] Regina Barzilay,[3,4,9,*] and James J. Collins[1,2,5,8,9,11,*]

[1] Department of Biological Engineering, Synthetic Biology Center, Institute for Medical Engineering and Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[2] Broad Institute of MIT and Harvard, Cambridge, MA 02142, USA
[3] Machine Learning for Pharmaceutical Discovery and Synthesis Consortium, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[4] Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[5] Wyss Institute for Biologically Inspired Engineering, Harvard University, Boston, MA 02115, USA
[6] Department of Biochemistry and Biomedical Sciences, Michael G. DeGroote Institute for Infectious Disease Research, McMaster University, Hamilton, ON L8N 3Z5, Canada
[7] Department of Genetics, Harvard Medical School, Boston, MA 02115, USA
[8] Harvard-MIT Program in Health Sciences and Technology, Cambridge, MA 02139, USA
[9] Abdul Latif Jameel Clinic for Machine Learning in Health, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[10] These authors contributed equally
[11] Lead Contact
*Correspondence: regina@csail.mit.edu (R.B.), jimjc@mit.edu (J.J.C.)

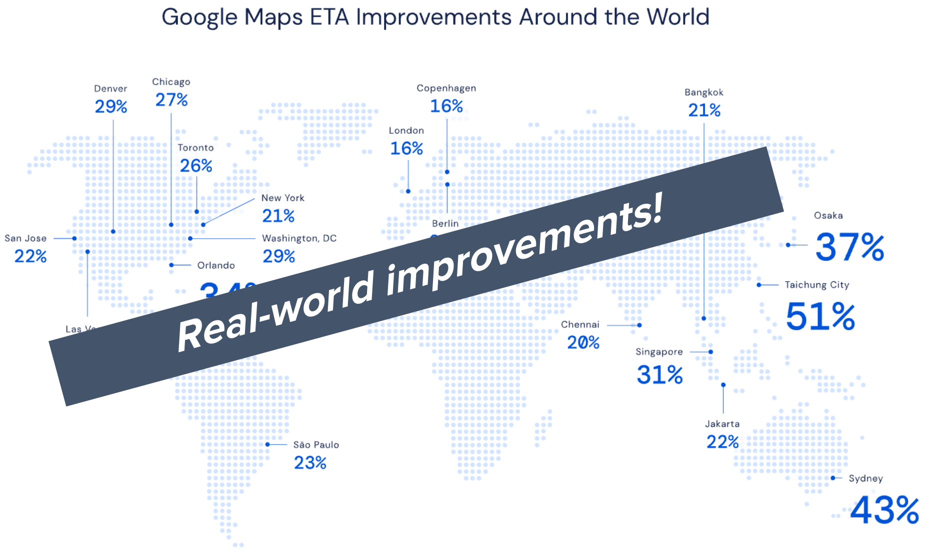
https://doi.org/10.1016/j.cell.2020.01.021

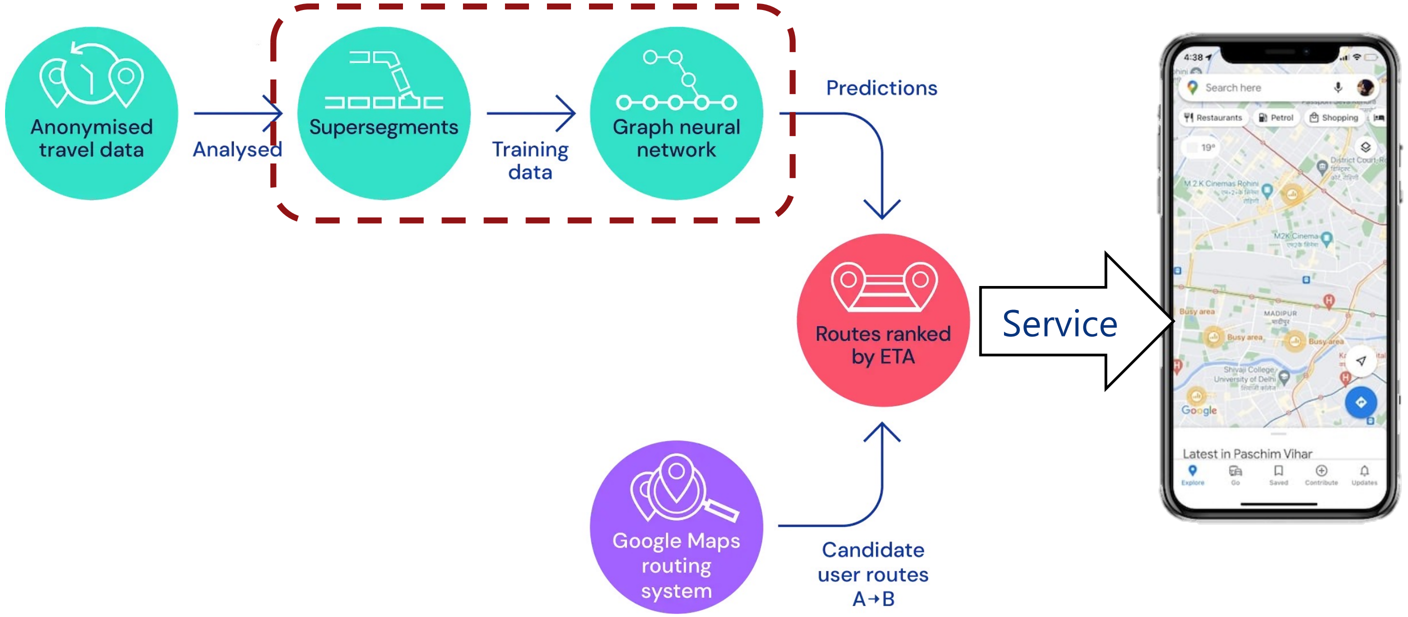


Stokes, Jonathan M., et al. "A deep learning approach to antibiotic discovery." Cell 180.4 (2020): 688-702.
Yang, Kevin, et al. "Analyzing learned molecular representations for property prediction." Journal of chemical information and modeling 59.8 (2019): 3370-3388.

Area 2) ETA prediction

Example 2: DeepMind’s improvement of Google map’s ETA (Estimated Time of Arrival) prediction




Anonymised travel data

Analysed

Supersegments

Training data

Graph neural network

Predictions

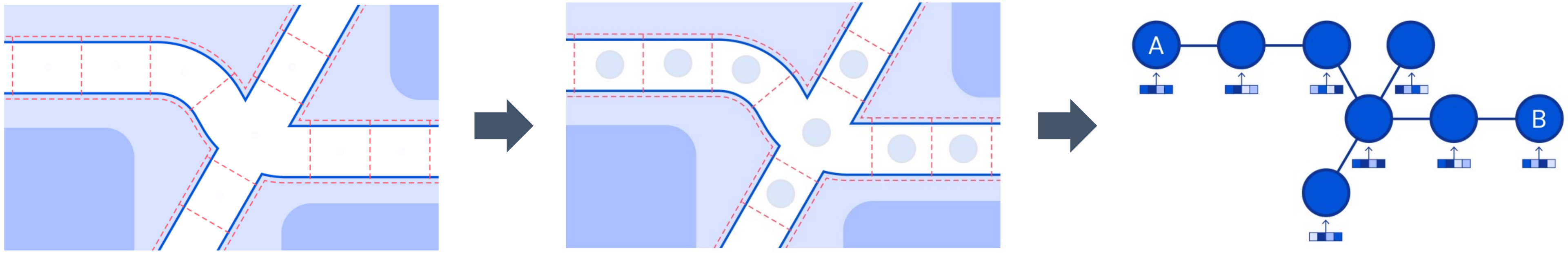
Routes ranked by ETA

Service

Google Maps routing system

Candidate user routes A→B

Unlike chemical datasets, constructing a graph is less straightforward.
In these cases, **how to construct the graph** is also a crucial contribution.

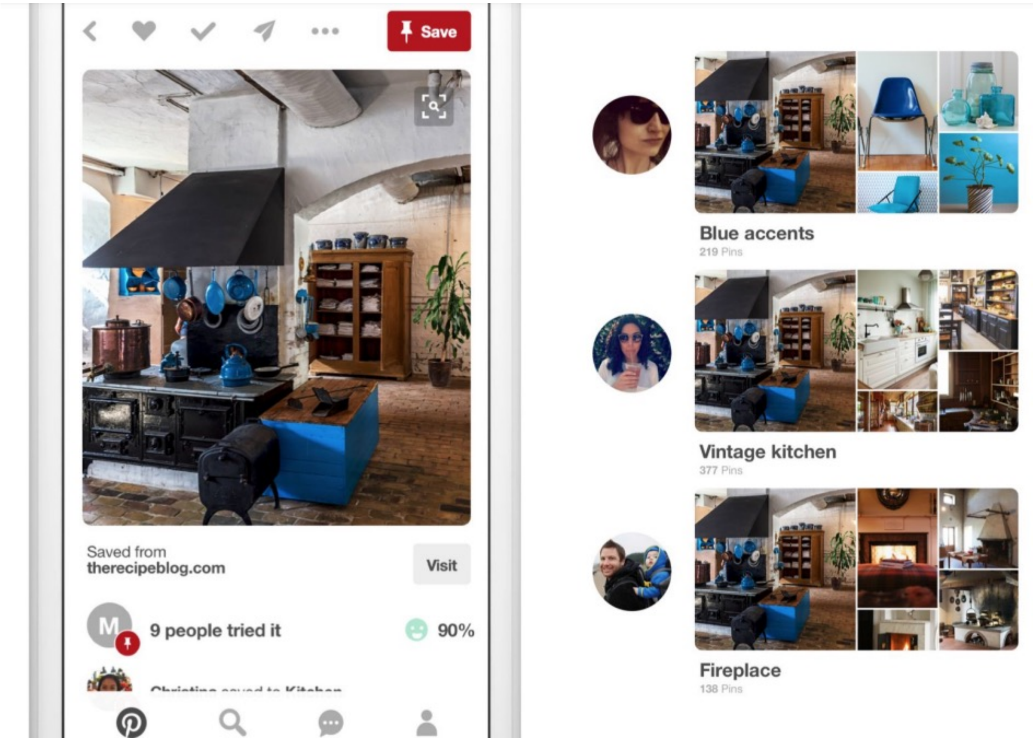


Derrow-Pinion, Austin, et al. "ETA prediction with graph neural networks in google maps.“, CIKM 2021.
Deepmind, “Traffic prediction with advanced graph neural networks”

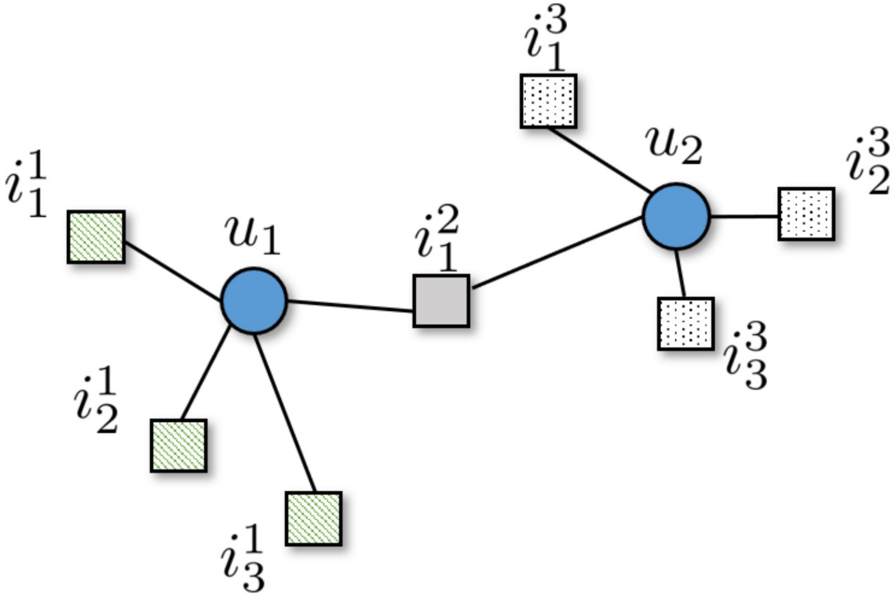
Area 3) Recommdender systems

Example 3: Pinterest (social platform)

Image & User interaction in Pinterest



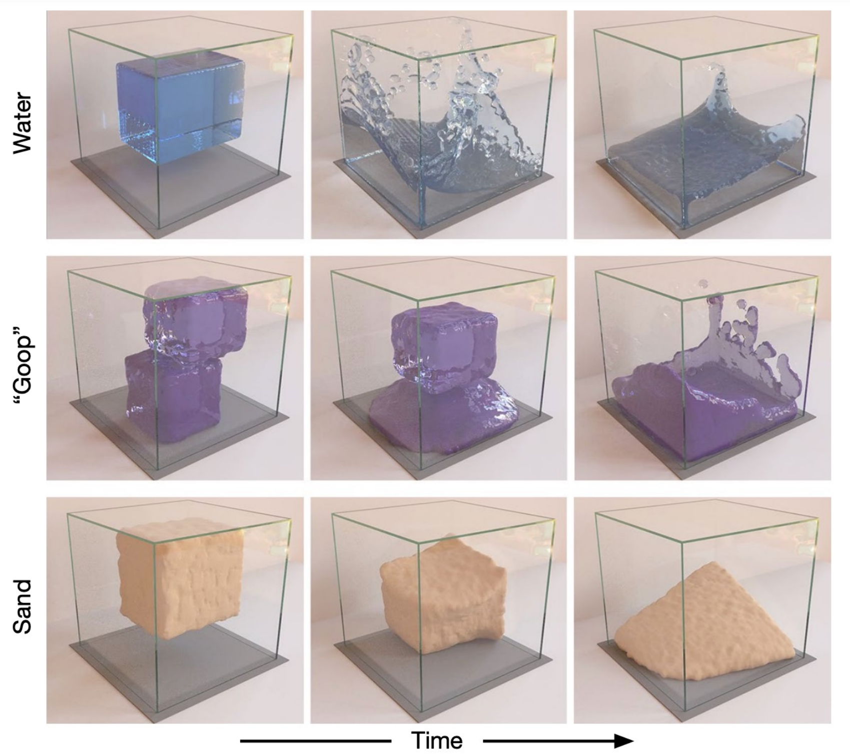
User-item interaction graph

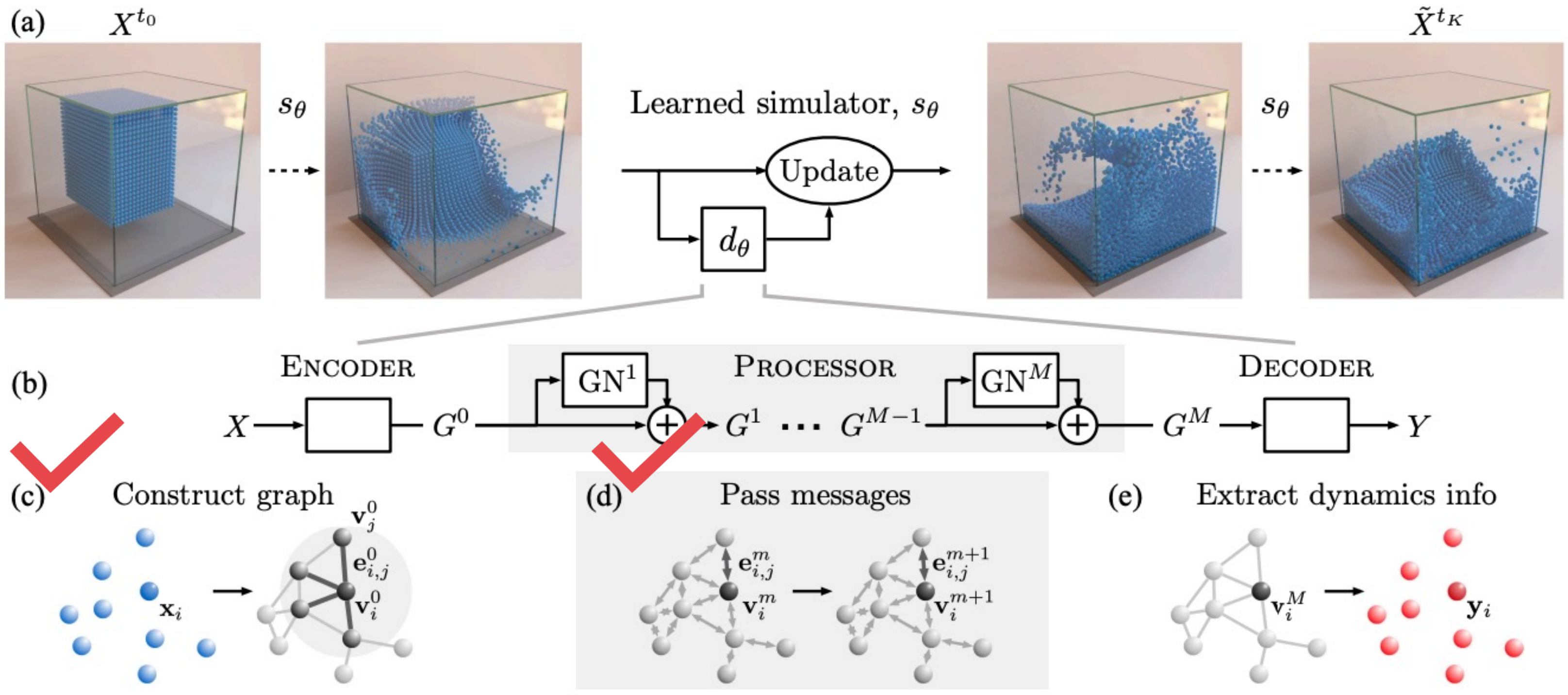


Source: Andrew Zhai (Pinterest) talk @WWW 2022 (link)
Right figure: Hou et al., Collaborative Filtering Based on Diffusion Models: Unveiling the Potential of High-Order Connectivity, SIGIR 2024

Area 4) Modeling physical systems

Example 4: Simulation of complex physical systems





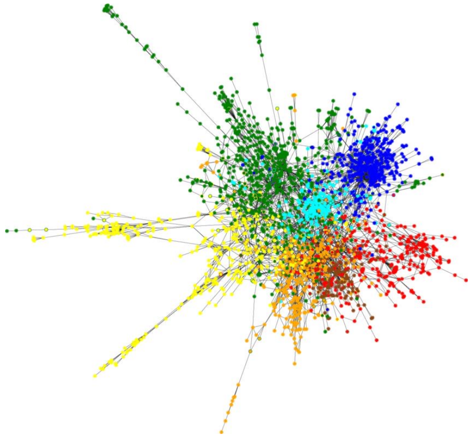
Sanchez-Gonzalez et al., Learning to Simulate Complex Physics with Graph Networks, ICML 2020

In academia: Benchmark datasets in the literature

Social

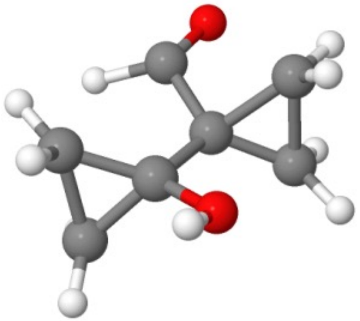
Node: People / Account
Edge: Connection
Node feature: Metadata

Citation / Web



Node: Paper
Edge: Citation
Node feature: Abstract

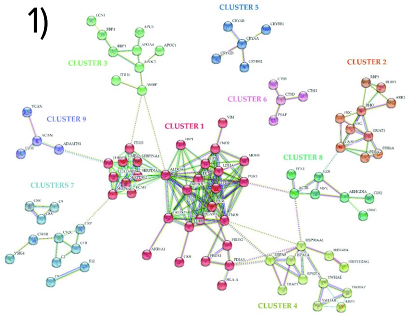
Molecules



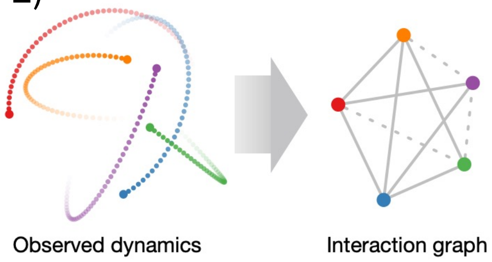
Node: Atom
Edge: Bond
Node feature: Atom type
Edge feature: Bond type

Biology / Simulation / etc.

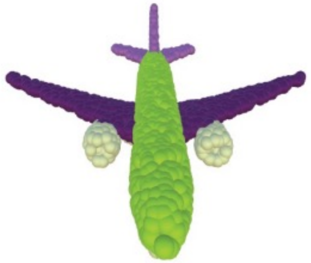
1)



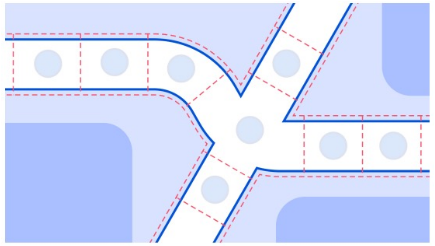
2)



3)



4)



Example benchmark datasets

- Reddit
- Ego-Facebook
- Github

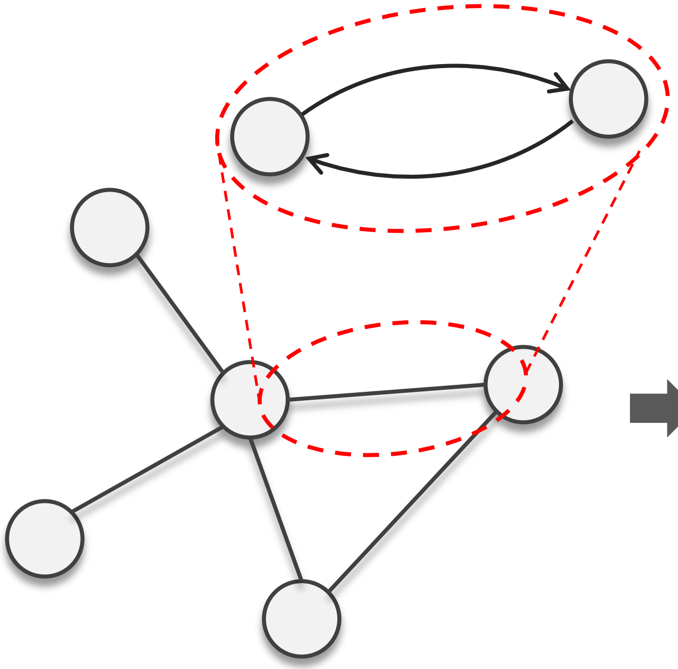
- *Planetoid dataset (Cora/Citeseer/Pubmed)
- Coauthor
- WebKB (Texas/Cornell/etc.)

- QM9
- Zinc
- MUTAG

1) **PPI (protein-protein interaction)
2) Physical simulation (Kipf et al., 2018)
3) 3D point cloud (Wang et al., 2019)
4) Road network (Derrow-Pinion et al., 2021)

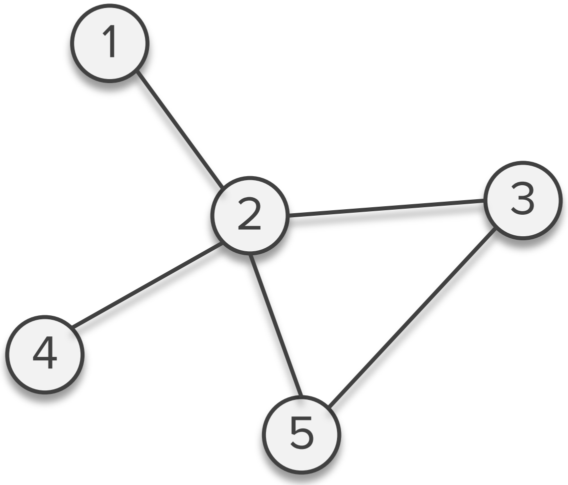
Yang et al., Revisiting Semi-Supervised Learning with Graph Embeddings, ICML 2016
Kipf et al., Neural Relational Inference for Interacting Systems, ICML 2018
Wang et al., Dynamic Graph CNN for Learning on Point Clouds, ACM Transactions on Graphics 2019
Derrow-Pinion et al., ETA Prediction with Graph Neural Networks in Google Maps, CIKM 2021
**Image source: https://www.researchgate.net/publication/324457787_iTRAQ_Quantitative_Proteomic_Analysis_of_Vitreous_from_Patients_with_Retinal_Detachment/figures?lo=1

Representing the graph as a adjacency matrix



*We treat undirected edges as two directed edges going in both directions

Undirected graph



Assign <u>arbitrary</u> node ordering

- **Graphs with canonical node ordering is not common**
- Related research topic: Positional encoding of nodes (As an example, see [1])

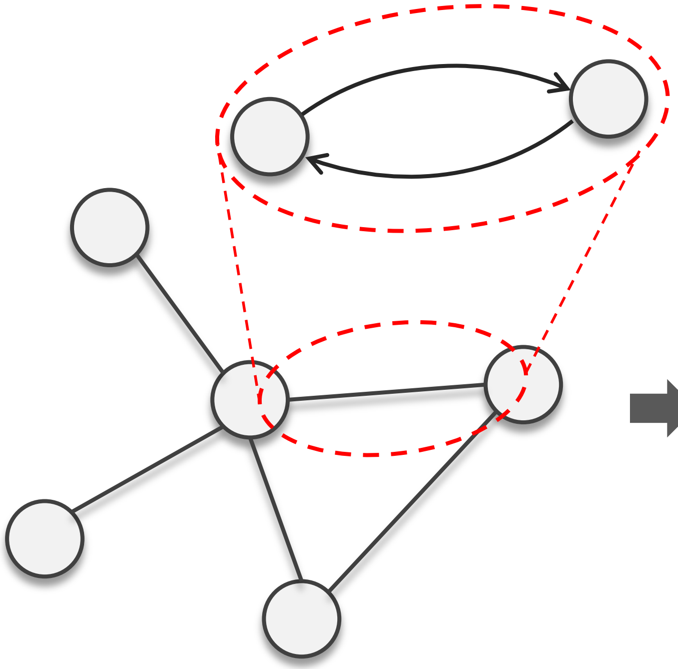
	1	2	3	4	5
1	0	1	0	0	0
2	1	0	1	1	1
3	0	1	0	0	1
4	0	1	0	0	0
5	0	1	0	1	0

Adjacency matrix

- Represent edge by assigning 1 for (i, j)-th element if node i and j are connected
- For <u>weighted</u> graphs: Assign a real number
- For graphs with <u>multiple</u> edges: Assign integers
- For <u>directed</u> graphs: Asymmetric matrix

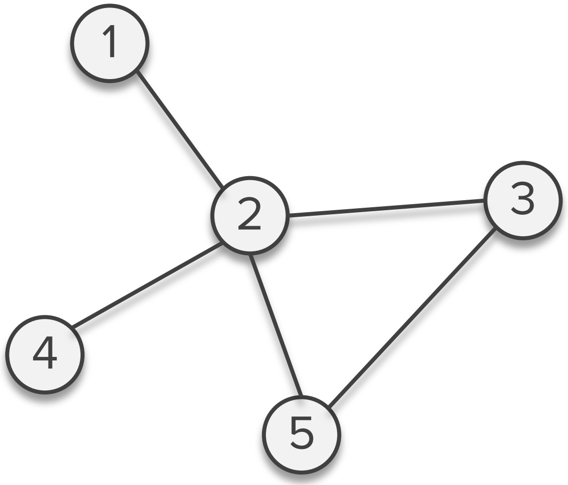
Maskey et al., Generalized Laplacian Positional Encoding for Graph Representation Learning, NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations

Representing the graph as a adjacency matrix



*We treat undirected edges as
two directed edges going in both directions

Undirected graph



Assign <u>arbitrary</u> node ordering

- **Graphs with canonical node ordering is not common**
- Related research topic: Positional encoding of nodes (As an example, see [1])

(1, 2), (2, 1), (2, 3), (3, 2), …

Edge list

- Represent graph by listing all edges
- Notice that for undirected edges, (i, j) and (j,i) both appear
- More **memory efficient** than (dense) adjacency matrix

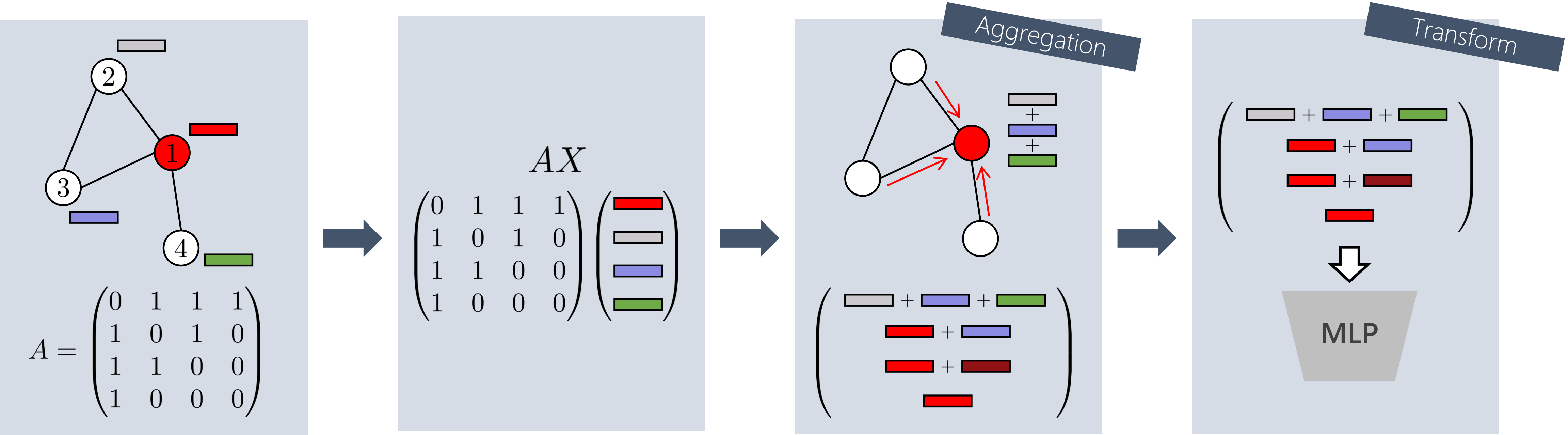
Maskey et al., Generalized Laplacian Positional Encoding for Graph Representation Learning, NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations

Understanding of the general framework of graph neural networks (GNNs)

A simple, popular, and straightforward GNN

GCN (Graph Convolutional Network): Kipf & Welling, ICLR 2017

We are now ready to understand the basic principles of GNN, by looking at the most popular architecture.


2
1
3
4
$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$
AX
$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$
Aggregation
+
+
Transform
+
MLP

Notice that, this whole procedure can be neatly expressed as: $\sigma(AX\Theta)$

Of course, all of this still holds **when we scramble the node ordering** (permutation invariant)

Non-linear activation function $\sigma(\cdot)$

Adjacency matrix $A \in \mathbb{R}^{n \times n}$

Node feature matrix $X \in \mathbb{R}^{n \times d}$

Learnable matrix $\Theta \in \mathbb{R}^{d \times d'}$

n: # of nodes

d: node feature dimensions

d': dimension for the next layer

A simple, popular, and straightforward GNN

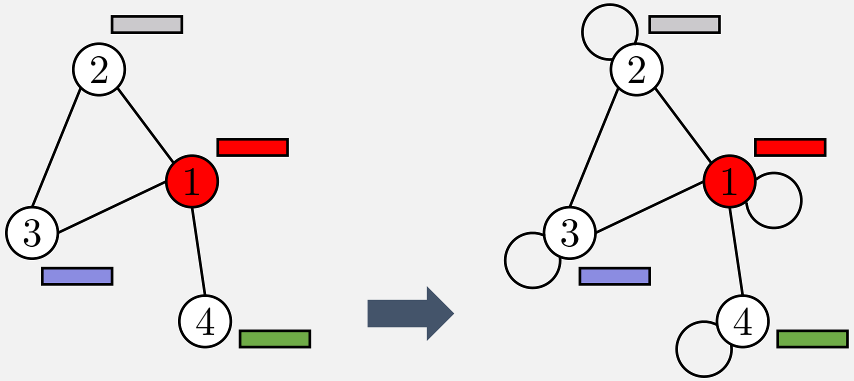
GCN (Graph Convolutional Network): Kipf & Welling, ICLR 2017

Of course, we can get creative with the graph structure to solve some practical issues

Problem 1: The information of the neighbor nodes are aggregated but not the node itself.
Problem 2: The adjacency matrix is not normalized, and the scale of the feature vectors may explode for repeated layers.

Resolution to problem 1

Add self-loops to each node

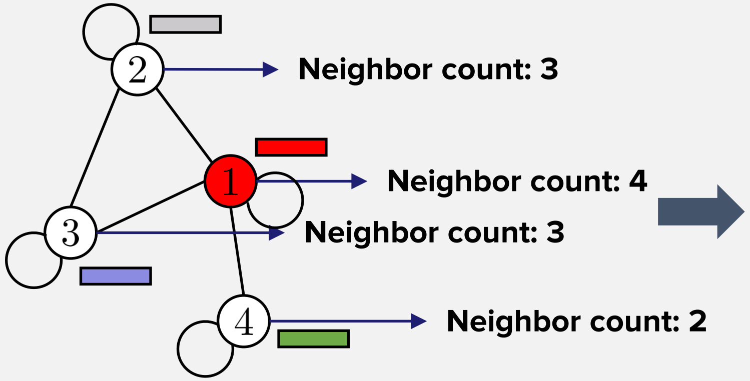


$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\hat{A} = A + I = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Resolution to problem 2

Normalization of $\hat{A}$:



Neighbor count: 3

Neighbor count: 4

Neighbor count: 3

Neighbor count: 2

$$D = \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

$$\tilde{A} = \hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2} = \begin{pmatrix} \frac{1}{4} & \frac{1}{\sqrt{12}} & \frac{1}{\sqrt{12}} & \frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{12}} & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{\sqrt{12}} & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{\sqrt{8}} & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

Final layer of GCN: $\sigma(\tilde{A} X \Theta)$

Abstraction: A general message-passing layer of GNNs

1. Message passing phase (Aggregation)

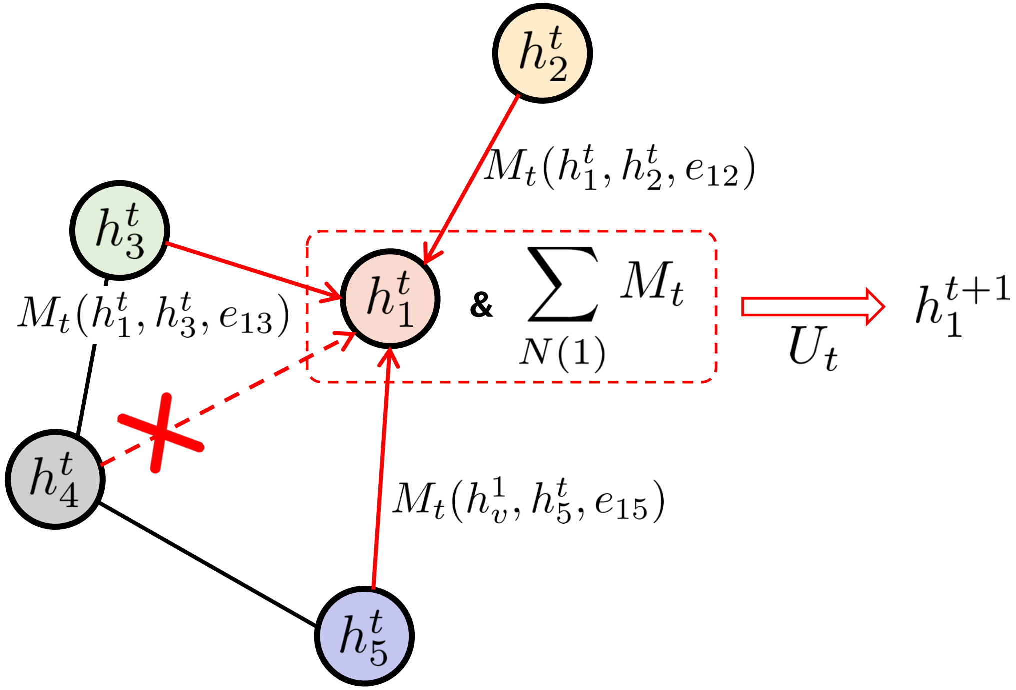
$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

2. Update phase (Transformation)

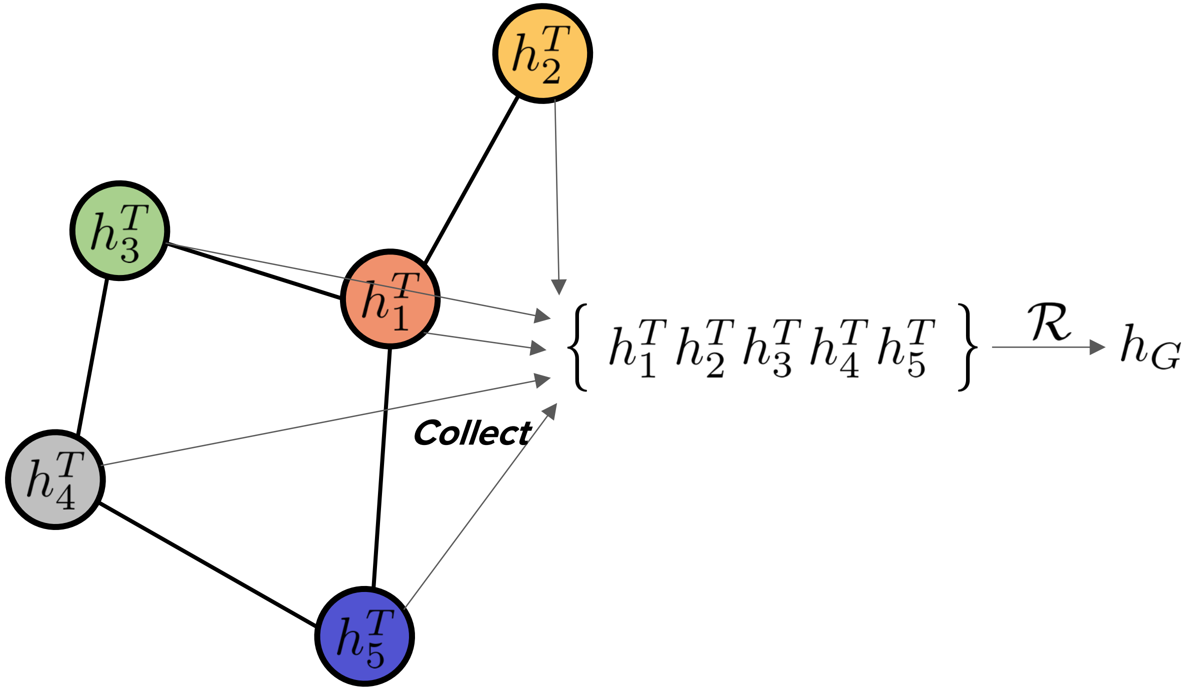
$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

3. Readout phase (Only for graph-level tasks)

$$h_G = \mathcal{R}(h_1^T, \cdots, h_{\mathcal{V}}^T)$$



h_2^t
$M_t(h_1^t, h_2^t, e_{12})$
h_3^t
$M_t(h_1^t, h_3^t, e_{13})$
h_1^t
&
$\sum_{N(1)} M_t$
U_t
h_1^{t+1}
h_4^t
$M_t(h_v^1, h_5^t, e_{15})$
h_5^t



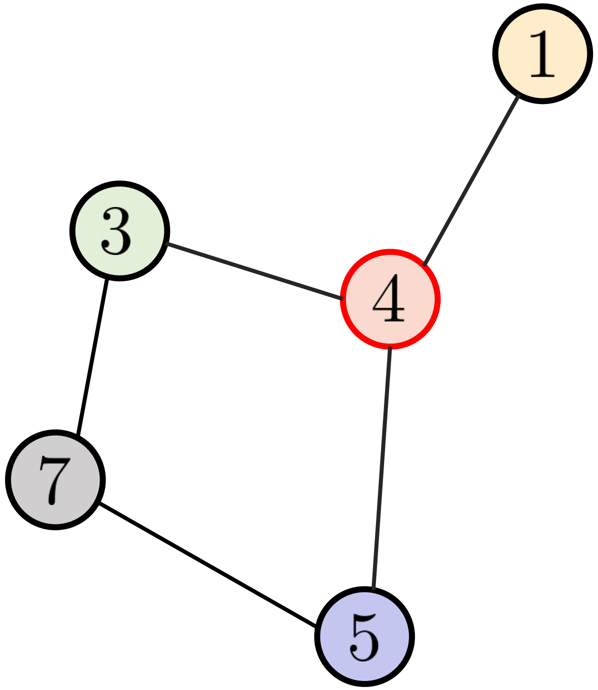
*Usually, we cite these papers for the term "message-passing"

[First formal introduction of the concept] Gilmer et al., "Neural Message Passing for Quantum Chemistry", ICML 2017

[Comprehensive discussion & abstraction] Bronstein et al., Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, arXiv 2021

Abstraction: A general message-passing layer of GNNs

GNN layer (Message-passing neural networks)



$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right)$$

$$\bigoplus$$

This operation must be permutation invariant to ensure the same result for different node orderings! **Summation / Average / Max pooling** etc.

So if we re-describe GCN for node 4, it would be…

$$\mathcal{N}_u = \{1, 3, 5\} \cup \{4\} \quad \psi(\mathbf{x}_u, \mathbf{x}_1) = \frac{1}{\sqrt{2 \times 4}}\mathbf{x}_1 \quad \phi = \mathrm{MLP}$$

*Usually, we cite these papers for the term "message-passing"
[First formal introduction of the concept] Gilmer et al., "Neural Message Passing for Quantum Chemistry", ICML 2017
[Comprehensive discussion & abstraction] Bronstein et al., Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, arXiv 2021

High-level understanding of several key GNN architectures

A list of noteworthy GNN architectures

Frequently used architectures (Must know!)

GCN) Kipf & Welling, “Semi-supervised classification with graph convolutional networks”, ICLR 2017

GraphSAGE) Hamilton et al., “Inductive representation learning on large graphs”, NeurIPS 2017

GAT) Veličković et al., “Graph attention networks”, ICLR 2018

GIN) Xu et al., “How powerful are graph neural networks?”, ICLR 2019 (we will come back to this in later seminars)

Lightweight GNNs (we will come back to this in later seminars)

SGC) Wu et al., “Simplifying graph convolutional networks”, ICML 2019

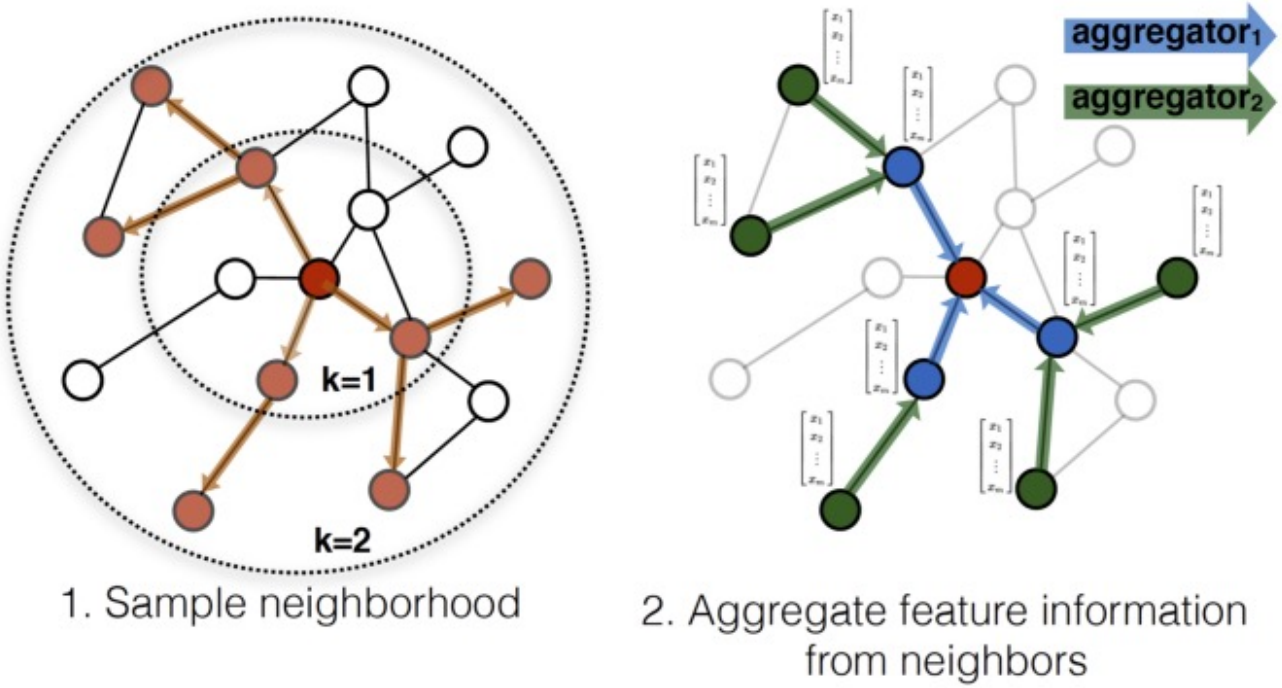
LightGCN) He et al., “LightGCN: Simplifying and powering graph convolutional network for recommendation, SIGIR 2020

Spectral viewpoint of GNNs (we will come back to this in later seminars)

ChebNet) Defferrard et al., “Convolutional neural networks on graphs with fast localized spectral filtering”,
NeurIPS 2016

GraphSAGE: Introduction of neighbor sampling

Problem: As we stack multiple layers, we introduce a **LOT** of neighboring nodes during message-passing.



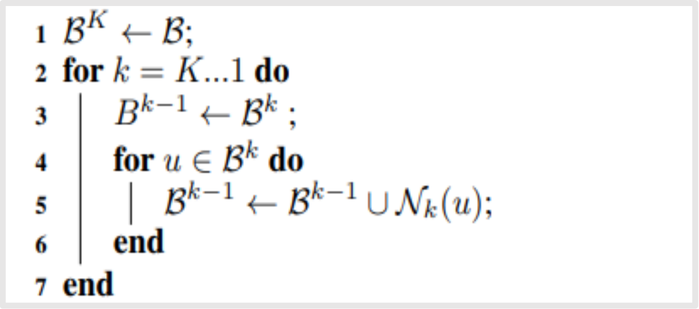
1. Sample neighborhood

2. Aggregate feature information from neighbors

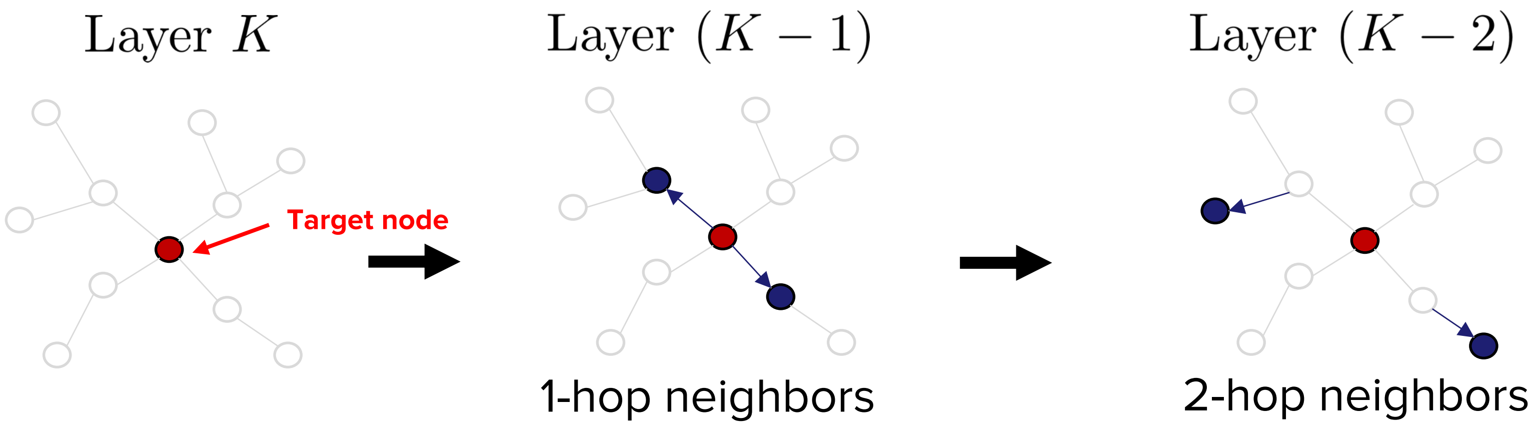
Sampling the neighbor nodes (contrast to using all neighbors) reduce memory complexity and still achieves good performance.

*In my experience, these type of intuitions (trading off speed and/or memory by dropping some nodes/edges) work better for social graph types

GraphSAGE: Introduction of neighbor sampling

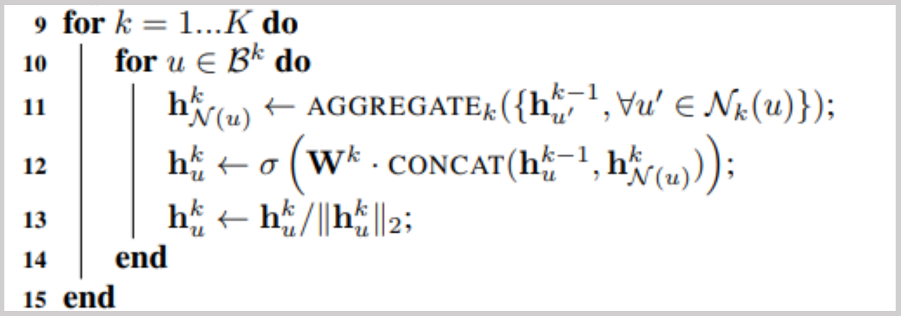


```
$\mathcal{B}^K \leftarrow \mathcal{B}$;
for $k = K...1$ do
    $\mathcal{B}^{k-1} \leftarrow \mathcal{B}^k$ ;
    for $u \in \mathcal{B}^k$ do
        $\mathcal{B}^{k-1} \leftarrow \mathcal{B}^{k-1} \cup \mathcal{N}_k(u)$;
    end
end
```

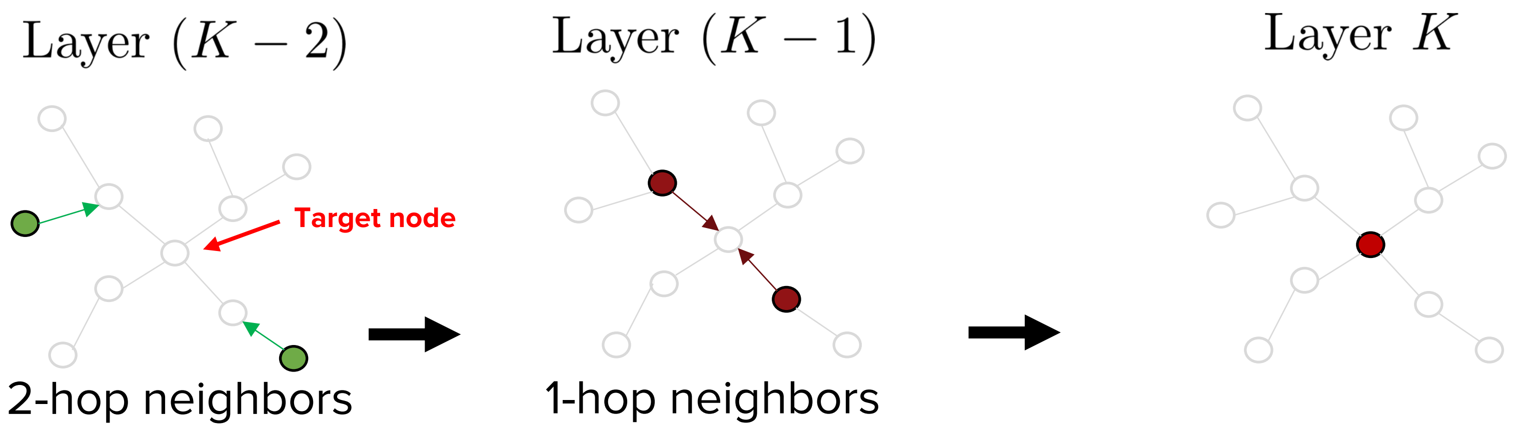


The sampling process is conceptually reversed compared to forward pass.

GraphSAGE: Introduction of neighbor sampling



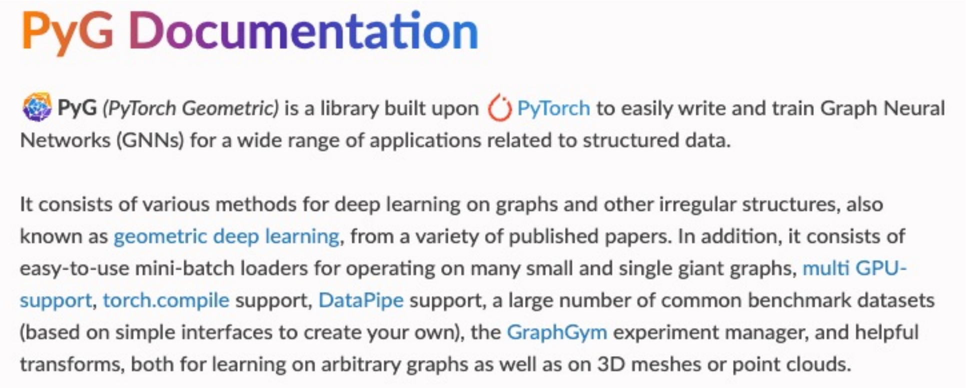
```
for $k = 1...K$ do
    for $u \in \mathcal{B}^k$ do
        $\mathbf{h}^k_{\mathcal{N}(u)} \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}^{k-1}_{u'}, \forall u' \in \mathcal{N}_k(u)\})$;
        $\mathbf{h}^k_u \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}^{k-1}_u, \mathbf{h}^k_{\mathcal{N}(u)})\right)$;
        $\mathbf{h}^k_u \leftarrow \mathbf{h}^k_u / \|\mathbf{h}^k_u\|_2$;
    end
end
```



The feed-forward process (message-passing) is conceptually reversed compared to forward pass.

Final note: Library for graph learning

PyTorch Geometric (link)



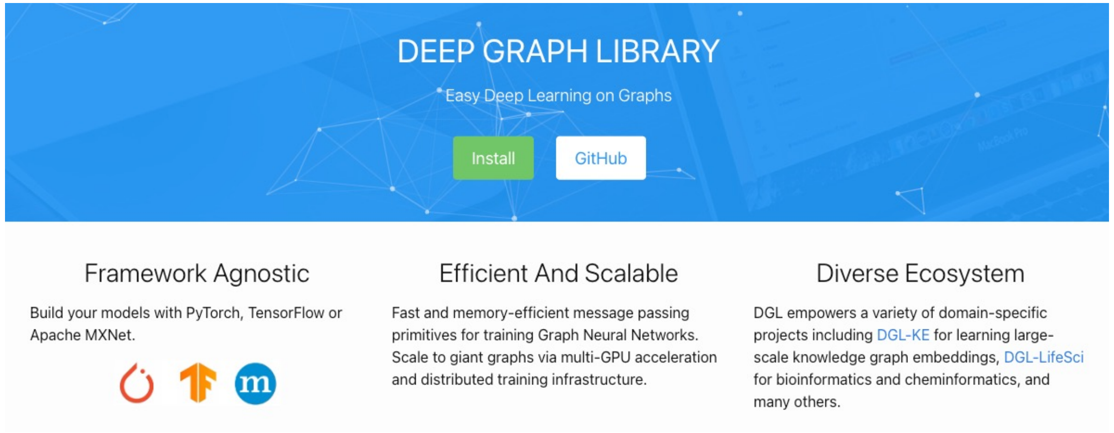
PyG Documentation

PyG (*PyTorch Geometric*) is a library built upon PyTorch to easily write and train Graph Neural Networks (GNNs) for a wide range of applications related to structured data.

It consists of various methods for deep learning on graphs and other irregular structures, also known as geometric deep learning, from a variety of published papers. In addition, it consists of easy-to-use mini-batch loaders for operating on many small and single giant graphs, multi GPU-support, torch.compile support, DataPipe support, a large number of common benchmark datasets (based on simple interfaces to create your own), the GraphGym experiment manager, and helpful transforms, both for learning on arbitrary graphs as well as on 3D meshes or point clouds.

- Jure Leskovec (Standford/KumoAI/Snapchat)
- Faster library updates (is this a good thing?)
- (Seems like) A larger community

Deep Graph Library (link)



DEEP GRAPH LIBRARY

Easy Deep Learning on Graphs

Framework Agnostic

Build your models with PyTorch, TensorFlow or Apache MXNet.

Efficient And Scalable

Fast and memory-efficient message passing primitives for training Graph Neural Networks. Scale to giant graphs via multi-GPU acceleration and distributed training infrastructure.

Diverse Ecosystem

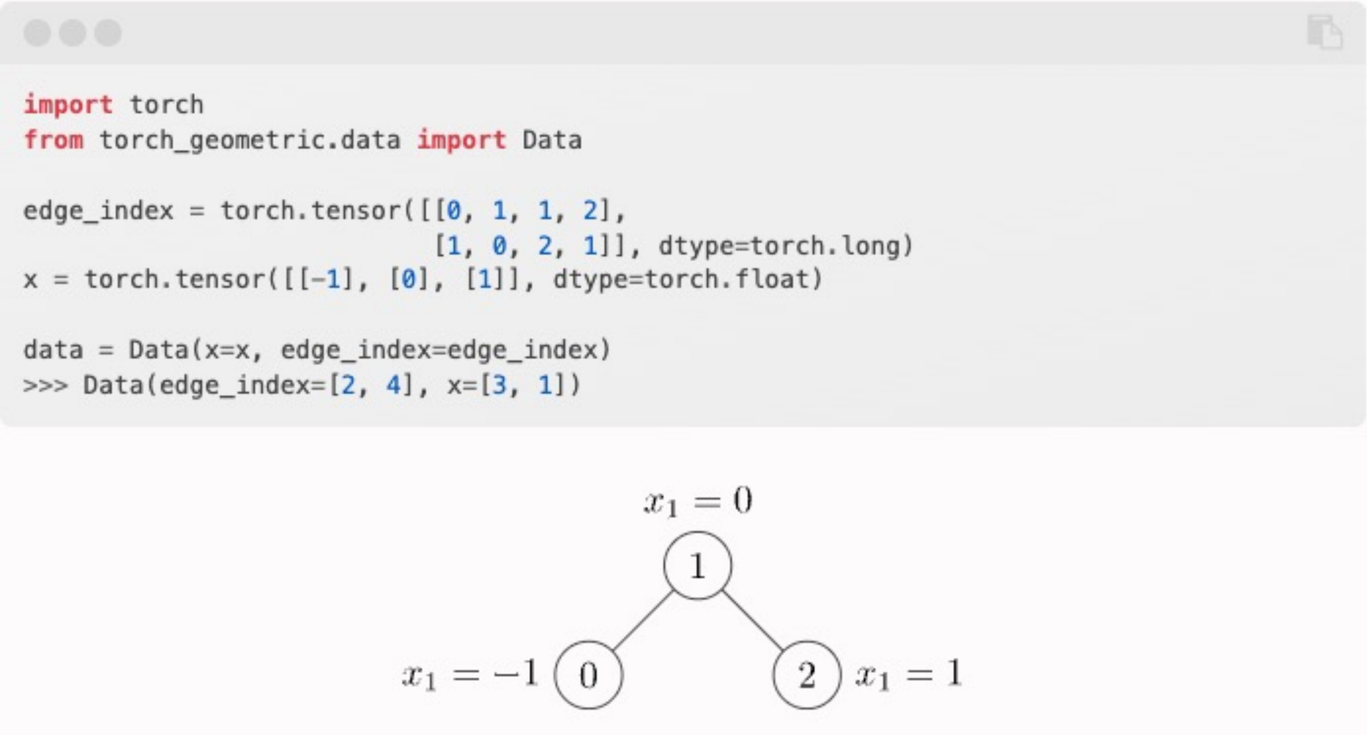
DGL empowers a variety of domain-specific projects including DGL-KE for learning large-scale knowledge graph embeddings, DGL-LifeSci for bioinformatics and cheminformatics, and many others.

- Slower library updates (is this a bad thing?)
- Variable framework support

NetworkX
Network Analysis in Python

- Additonal library: NetworkX (link) – Library for graphs in general
  - Not a library for ML/DL
  - Often used in junction with PyG/DGL

Final note: Library for graph learning

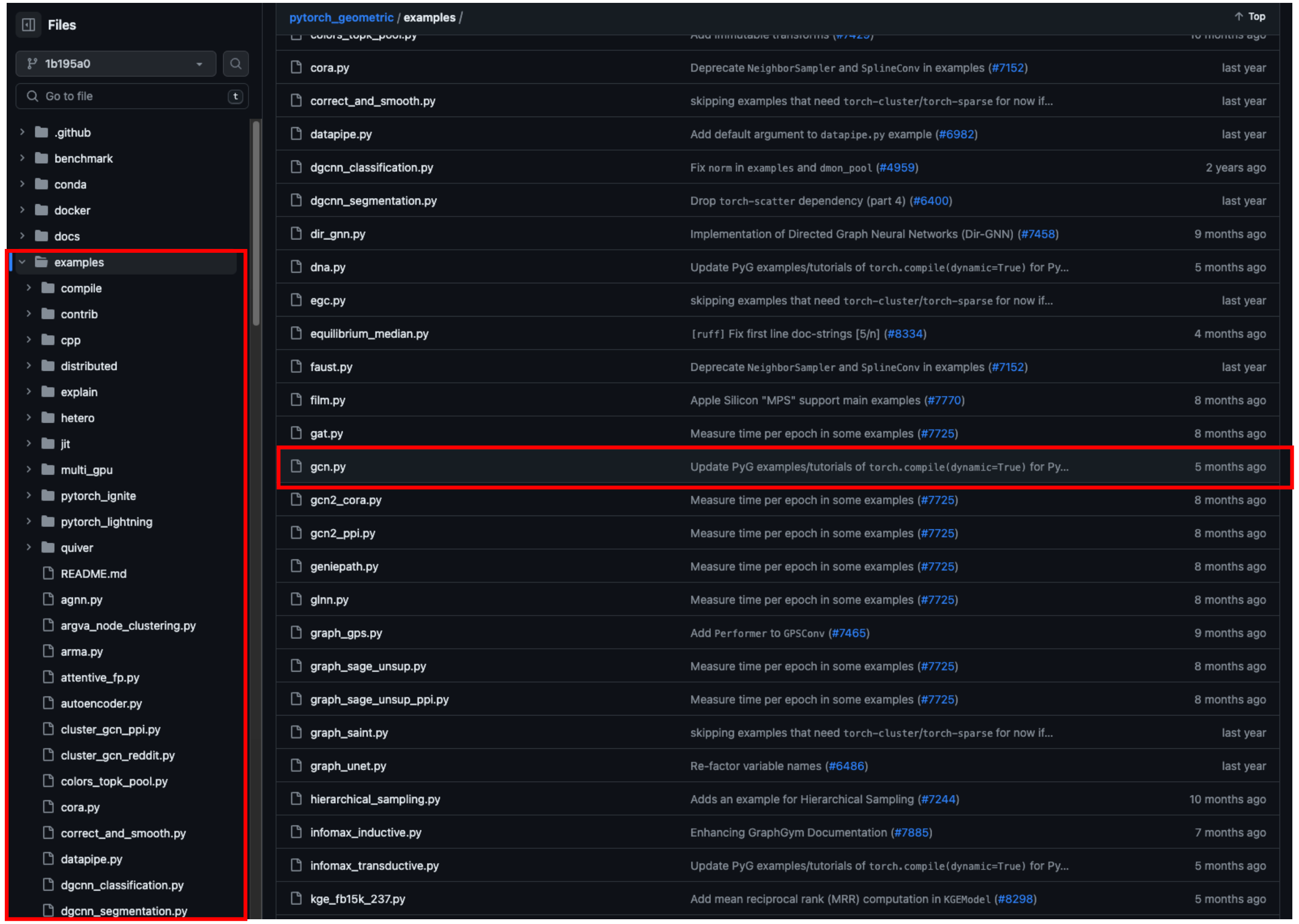


```
import torch
from torch_geometric.data import Data

edge_index = torch.tensor([[0, 1, 1, 2],
                           [1, 0, 2, 1]], dtype=torch.long)
x = torch.tensor([[-1], [0], [1]], dtype=torch.float)

data = Data(x=x, edge_index=edge_index)
>>> Data(edge_index=[2, 4], x=[3, 1])
```

- You *at minimum* need to define data.edge_index
- Node features are usually represented as data.x
- Don’t forget to include both directions for undirected graphs
- Most graph processing/manipulation tools are in torch_geometric.utils

Final note: Library for graph learning

Go to the examples folder in the repo



- .github
- benchmark
- conda
- docker
- docs
- examples
  - compile
  - contrib
  - cpp
  - distributed
  - explain
  - hetero
  - jit
  - multi_gpu
  - pytorch_ignite
  - pytorch_lightning
  - quiver
  - README.md
  - agnn.py
  - argva_node_clustering.py
  - arma.py
  - attentive_fp.py
  - autoencoder.py
  - cluster_gcn_ppi.py
  - cluster_gcn_reddit.py
  - colors_topk_pool.py
  - cora.py
  - correct_and_smooth.py
  - datapipe.py
  - dgcnn_classification.py
  - dgcnn_segmentation.py

pytorch_geometric / examples /

colors_topk_pool.py	Add immutable transforms (#7425)	10 months ago
cora.py	Deprecate `NeighborSampler` and `SplineConv` in examples (#7152)	last year
correct_and_smooth.py	skipping examples that need `torch-cluster/torch-sparse` for now if...	last year
datapipe.py	Add default argument to `datapipe.py` example (#6982)	last year
dgcnn_classification.py	Fix `norm` in `examples` and `dmon_pool` (#4959)	2 years ago
dgcnn_segmentation.py	Drop `torch-scatter` dependency (part 4) (#6400)	last year
dir_gnn.py	Implementation of Directed Graph Neural Networks (Dir-GNN) (#7458)	9 months ago
dna.py	Update PyG examples/tutorials of `torch.compile(dynamic=True)` for Py...	5 months ago
egc.py	skipping examples that need `torch-cluster/torch-sparse` for now if...	last year
equilibrium_median.py	`[ruff]` Fix first line doc-strings [5/n] (#8334)	4 months ago
faust.py	Deprecate `NeighborSampler` and `SplineConv` in examples (#7152)	last year
film.py	Apple Silicon "MPS" support main examples (#7770)	8 months ago
gat.py	Measure time per epoch in some examples (#7725)	8 months ago
gcn.py	Update PyG examples/tutorials of `torch.compile(dynamic=True)` for Py...	5 months ago
gcn2_cora.py	Measure time per epoch in some examples (#7725)	8 months ago
gcn2_ppi.py	Measure time per epoch in some examples (#7725)	8 months ago
geniepath.py	Measure time per epoch in some examples (#7725)	8 months ago
glnn.py	Measure time per epoch in some examples (#7725)	8 months ago
graph_gps.py	Add `Performer` to `GPSConv` (#7465)	9 months ago
graph_sage_unsup.py	Measure time per epoch in some examples (#7725)	8 months ago
graph_sage_unsup_ppi.py	Measure time per epoch in some examples (#7725)	8 months ago
graph_saint.py	skipping examples that need `torch-cluster/torch-sparse` for now if...	last year
graph_unet.py	Re-factor variable names (#6486)	last year
hierarchical_sampling.py	Adds an example for Hierarchical Sampling (#7244)	10 months ago
infomax_inductive.py	Enhancing GraphGym Documentation (#7885)	7 months ago
infomax_transductive.py	Update PyG examples/tutorials of `torch.compile(dynamic=True)` for Py...	5 months ago
kge_fb15k_237.py	Add mean reciprocal rank (MRR) computation in `KGEModel` (#8298)	5 months ago

If you want to know how to run GCN, go to the gcn.py file!

Includes…

- How to prep the data (with preprocessing, data splits etc.)
- How to define the model
- How to set up the training iteration
- How to measure performance

Takeaways

1. Graphs are entities (nodes) that are **connected** (edges)
2. A lot of problems can be formulated as a graph learning problem
3. Graph neural networks = **Message-passing framework (Aggregate + Transformation)**

Thank you!

Please feel free to ask any questions :)

jordan7186.github.io