Graph Neural Networks

Johannes Lutzeyer

Advanced Deep Learning: Lecture 3

January 21, 2025

Overview of Today's Lecture

1) A Brief Review of Graph Representation Learning;

Overview of Today's Lecture

1) A Brief Review of Graph Representation Learning;

2) Graph Neural Networks;

Overview of Today's Lecture

1) A Brief Review of Graph Representation Learning;

2) Graph Neural Networks;

3) Different Approaches to Message Passing Including Recent Research;

Overview of Today's Lecture

1) A Brief Review of Graph Representation Learning;

2) Graph Neural Networks;

3) Different Approaches to Message Passing Including Recent Research;

4) Recent Advances on the Robustness of Graph Neural Networks.

Graph Representation Learning

Overall Goal: Learn "informative" representations of graph structured data

Graph Representation Learning

Overall Goal: Learn "informative" representations of graph structured data

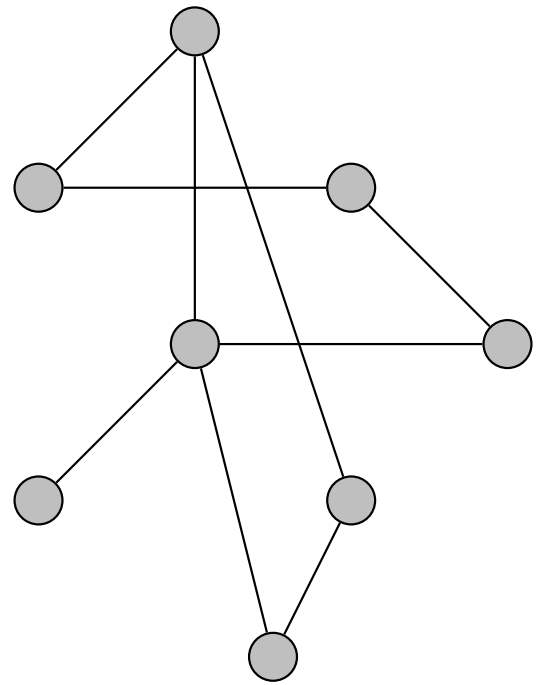
What is graph structured data?

Graph Representation Learning

Overall Goal: Learn "informative" representations of graph structured data

What is graph structured data?
It's the combination of

- a graph $G = (V, E)$;

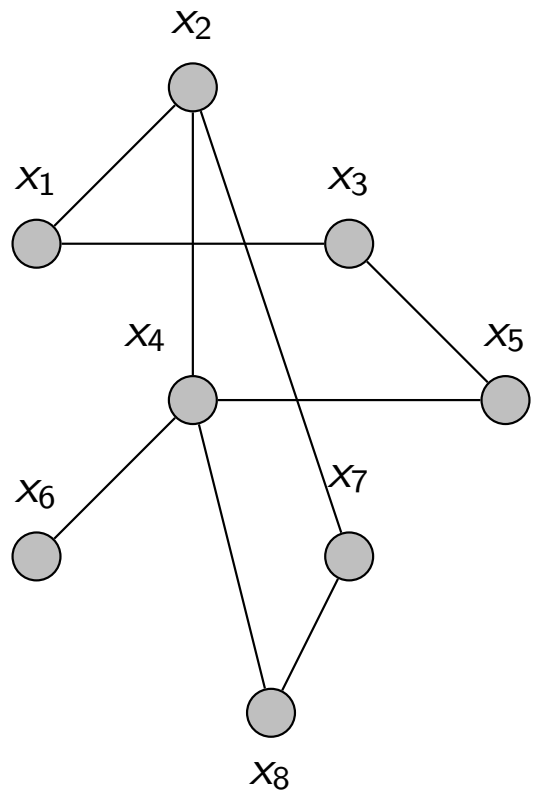


Graph Representation Learning

Overall Goal: Learn "informative" representations of graph structured data

What is graph structured data?
It's the combination of

- a graph $G = (V, E)$;
- node-features $X = [x_1, \ldots, x_n]^T$.

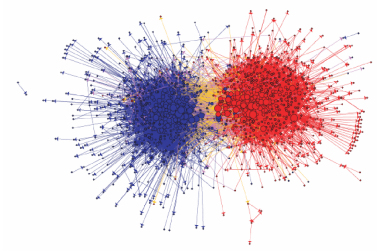


Graph Representation Learning

Overall Goal: Learn "informative" representations of graph structured data

What is graph structured data?
It's the combination of

- a graph $G = (V, E)$;
- node-features $X = [x_1, \ldots, x_n]^T$.



US political weblogs
(Adamic & Glance, 2005)

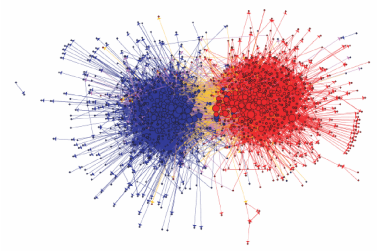
Where does it arise?

Graph Representation Learning

Overall Goal: Learn "informative" representations of graph structured data

What is graph structured data?
It's the combination of

- a graph $G = (V, E)$;
- node-features $X = [x_1, \ldots, x_n]^T$.



US political weblogs
(Adamic & Glance, 2005)

Where does it arise?

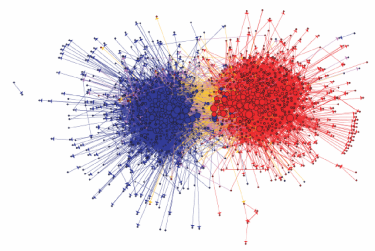
Caffeine molecule
(Bronstein, 2021)

Graph Representation Learning

Overall Goal: Learn "informative" representations of graph structured data

What is graph structured data?
It's the combination of

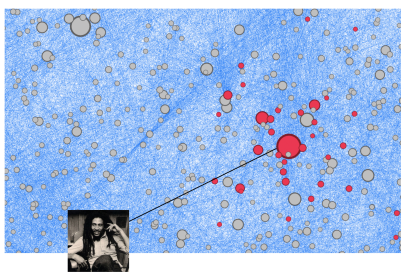
- a graph $G = (V, E)$;
- node-features $X = [x_1, \ldots, x_n]^T$.



US political weblogs
(Adamic & Glance, 2005)

Where does it arise?

Caffeine molecule
(Bronstein, 2021)



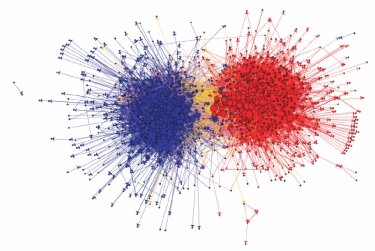
Deezer artists
(Salha-Galvan, 2022)

Graph Representation Learning

Overall Goal: Learn "informative" representations of graph structured data

What is graph structured data?
It's the combination of

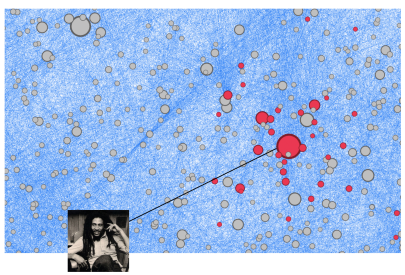
- a graph $G = (V, E)$;
- node-features $X = [x_1, \ldots, x_n]^T$.



US political weblogs
(Adamic & Glance, 2005)

Where does it arise?
It's ubiquitous!

Caffeine molecule
(Bronstein, 2021)



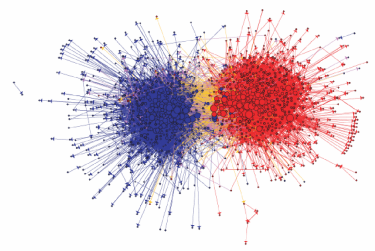
Deezer artists
(Salha-Galvan, 2022)

Graph Representation Learning

Overall Goal: Learn "informative" representations of graph structured data

What is graph structured data?
It's the combination of

- a graph $G = (V, E)$;
- node-features $X = [x_1, \ldots, x_n]^T$.

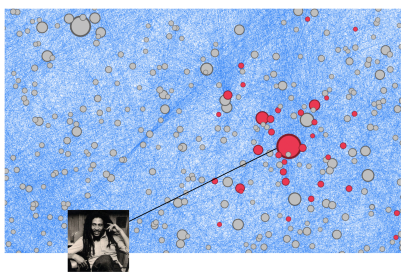


US political weblogs
(Adamic & Glance, 2005)

Where does it arise?
It's ubiquitous!

Caffeine molecule
(Bronstein, 2021)

What can we learn from it?



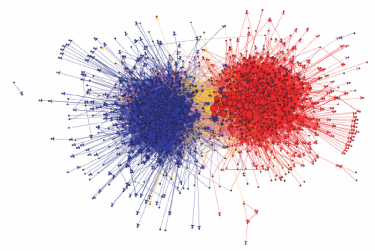
Deezer artists
(Salha-Galvan, 2022)

Graph Representation Learning

Overall Goal: Learn "informative" representations of graph structured data

What is graph structured data?
It's the combination of

- a graph $G = (V, E)$;
- node-features $X = [x_1, \ldots, x_n]^T$.



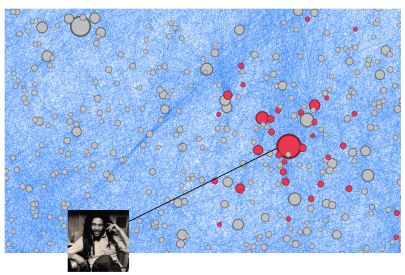
US political weblogs
(Adamic & Glance, 2005)

Where does it arise?
It's ubiquitous!

Caffeine molecule
(Bronstein, 2021)

What can we learn from it?

- Node and Graph Classification



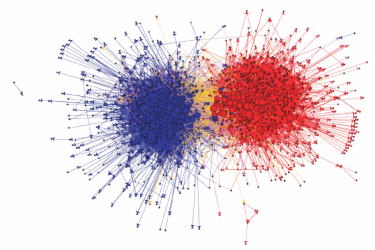
Deezer artists
(Salha-Galvan, 2022)

Graph Representation Learning

Overall Goal: Learn "informative" representations of graph structured data

What is graph structured data?
It's the combination of

- a graph $G = (V, E)$;
- node-features $X = [x_1, \ldots, x_n]^T$.



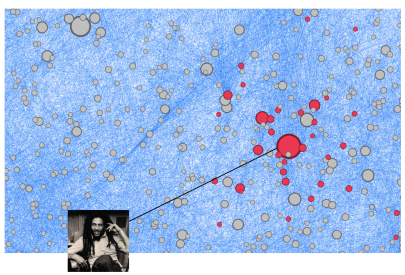
US political weblogs
(Adamic & Glance, 2005)

Where does it arise?
It's ubiquitous!

Caffeine molecule
(Bronstein, 2021)

What can we learn from it?

- Node and Graph Classification
- Node and Graph Regression



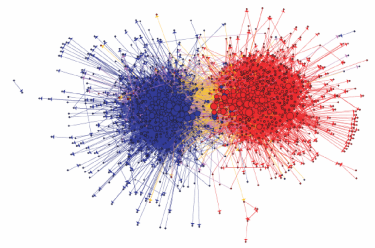
Deezer artists
(Salha-Galvan, 2022)

Graph Representation Learning

Overall Goal: Learn "informative" representations of graph structured data

What is graph structured data?
It's the combination of

- a graph $G = (V, E)$;
- node-features $X = [x_1, \ldots, x_n]^T$.



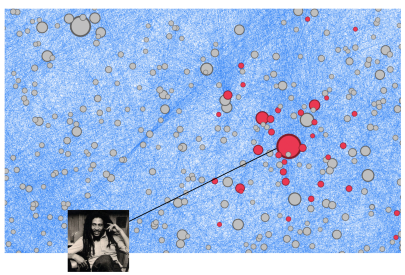
US political weblogs
(Adamic & Glance, 2005)

Where does it arise?
It's ubiquitous!

Caffeine molecule
(Bronstein, 2021)

What can we learn from it?

- Node and Graph Classification
- Node and Graph Regression
- Link Prediction



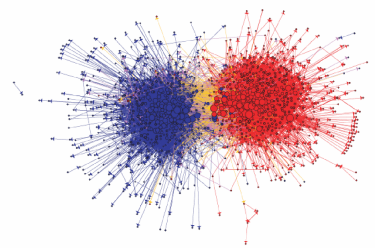
Deezer artists
(Salha-Galvan, 2022)

Graph Representation Learning

Overall Goal: Learn "informative" representations of graph structured data

What is graph structured data?
It's the combination of

- a graph $G = (V, E)$;
- node-features $X = [x_1, \ldots, x_n]^T$.



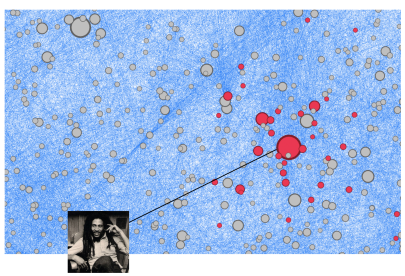
US political weblogs
(Adamic & Glance, 2005)

Where does it arise?
It's ubiquitous!

Caffeine molecule
(Bronstein, 2021)

What can we learn from it?

- Node and Graph Classification
- Node and Graph Regression
- Link Prediction
- Can you think of another?



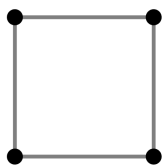
Deezer artists
(Salha-Galvan, 2022)

Graph Shift Operators

Definition

Graphs $G = (V, E)$ can be represented using:

- *adjacency matrix* $A \in \{0,1\}^{n \times n}$ where $A_{ij} = 1$ iff $(i,j) \in E$.



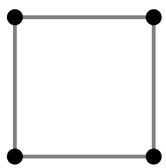
$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Graph Shift Operators

Definition

Graphs $G = (V, E)$ can be represented using:

- *adjacency matrix* $A \in \{0,1\}^{n \times n}$ where $A_{ij} = 1$ iff $(i,j) \in E$.
- *unnormalised graph Laplacian matrix* $L = D - A$, where $D = \mathrm{diag}(A\mathbf{1}_n)$.



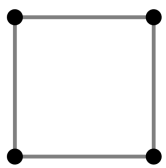
$$A = \begin{pmatrix} 0&1&0&1 \\ 1&0&1&0 \\ 0&1&0&1 \\ 1&0&1&0 \end{pmatrix} \qquad L = \begin{pmatrix} 2&-1&0&-1 \\ -1&2&-1&0 \\ 0&-1&2&-1 \\ -1&0&-1&2 \end{pmatrix}$$

Graph Shift Operators

Definition

Graphs $G = (V, E)$ can be represented using:

- *adjacency matrix* $A \in \{0,1\}^{n \times n}$ where $A_{ij} = 1$ iff $(i,j) \in E$.
- *unnormalised graph Laplacian matrix* $L = D - A$, where $D = \mathrm{diag}(A\mathbf{1}_n)$.
- *symmetric normalised graph Laplacian matrix* $L_{sym} = D^{-1/2} L D^{-1/2}$ and *random-walk normalised Laplacian matrix* $L_{rw} = D^{-1} L$.



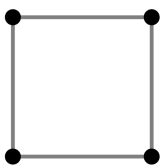
$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad L = \begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix} \quad L_{sym} = \begin{pmatrix} 1 & -0.5 & 0 & -0.5 \\ -0.5 & 1 & -0.5 & 0 \\ 0 & -0.5 & 1 & -0.5 \\ -0.5 & 0 & -0.5 & 1 \end{pmatrix}$$

Graph Shift Operators

Definition

Graphs $G = (V, E)$ can be represented using:

- *adjacency matrix* $A \in \{0,1\}^{n \times n}$ where $A_{ij} = 1$ iff $(i,j) \in E$.
- *unnormalised graph Laplacian matrix* $L = D - A$, where $D = \mathrm{diag}(A\mathbf{1}_n)$.
- *symmetric normalised graph Laplacian matrix* $L_{sym} = D^{-1/2} L D^{-1/2}$ and *random-walk normalised Laplacian matrix* $L_{rw} = D^{-1} L$.



$$A = \begin{pmatrix} 0 & 1 & \mathbf{0} & 1 \\ 1 & 0 & 1 & \mathbf{0} \\ \mathbf{0} & 1 & 0 & 1 \\ 1 & \mathbf{0} & 1 & 0 \end{pmatrix} \quad L = \begin{pmatrix} 2 & -1 & \mathbf{0} & -1 \\ -1 & 2 & -1 & \mathbf{0} \\ \mathbf{0} & -1 & 2 & -1 \\ -1 & \mathbf{0} & -1 & 2 \end{pmatrix} \quad L_{sym} = \begin{pmatrix} 1 & -0.5 & \mathbf{0} & -0.5 \\ -0.5 & 1 & -0.5 & \mathbf{0} \\ \mathbf{0} & -0.5 & 1 & -0.5 \\ -0.5 & \mathbf{0} & -0.5 & 1 \end{pmatrix}$$

Definition (Sandryhaila and Moura, 2013)

A matrix $S \in \mathbb{R}^{n \times n}$ is called a *Graph Shift Operator* (GSO) if it satisfies: $S_{ij} = 0$ for $i \neq j$ and $(i,j) \notin E$.

Multi Layer Perceptrons

Multi Layer Perceptrons (MLPs) are neural networks that take vector-valued data as input.

Multi Layer Perceptrons

Multi Layer Perceptrons (MLPs) are neural networks that take vector-valued data as input.

The model equation of an MLP layer is

$$h_i^{(k)} = \sigma\left(h_i^{(k-1)} W\right),$$

where

- $h_i^{(k)} \in \mathbb{R}^d$ denotes the hidden state learned to represent data point i at layer k of the MLP,

Multi Layer Perceptrons

Multi Layer Perceptrons (MLPs) are neural networks that take vector-valued data as input.

The model equation of an MLP layer is

$$h_i^{(k)} = \sigma\left(h_i^{(k-1)} W\right),$$

where

- $h_i^{(k)} \in \mathbb{R}^d$ denotes the hidden state learned to represent data point i at layer k of the MLP,
- the function $\sigma(\cdot)$ is an activation function, such as ReLU, that is defined to act elementwise on vector-valued inputs,

Multi Layer Perceptrons

Multi Layer Perceptrons (MLPs) are neural networks that take vector-valued data as input.

The model equation of an MLP layer is

$$h_i^{(k)} = \sigma\left(h_i^{(k-1)} W\right),$$

where

- $h_i^{(k)} \in \mathbb{R}^d$ denotes the hidden state learned to represent data point i at layer k of the MLP,
- the function $\sigma(\cdot)$ is an activation function, such as ReLU, that is defined to act elementwise on vector-valued inputs,
- $W \in \mathbb{R}^{d' \times d}$ is a trainable weight matrix,

Multi Layer Perceptrons

Multi Layer Perceptrons (MLPs) are neural networks that take vector-valued data as input.

The model equation of an MLP layer is

$$h_i^{(k)} = \sigma\left(h_i^{(k-1)} W\right),$$

where

- $h_i^{(k)} \in \mathbb{R}^d$ denotes the hidden state learned to represent data point i at layer k of the MLP,
- the function $\sigma(\cdot)$ is an activation function, such as ReLU, that is defined to act elementwise on vector-valued inputs,
- $W \in \mathbb{R}^{d' \times d}$ is a trainable weight matrix,
- note that the trainable bias vector of an MLP is omitted here for brevity.

Multi Layer Perceptrons

Multi Layer Perceptrons (MLPs) are neural networks that take vector-valued data as input.

The model equation of an MLP layer is

$$h_i^{(k)} = \sigma\left(h_i^{(k-1)} W\right),$$

where

- $h_i^{(k)} \in \mathbb{R}^d$ denotes the hidden state learned to represent data point i at layer k of the MLP,
- the function $\sigma(\cdot)$ is an activation function, such as ReLU, that is defined to act elementwise on vector-valued inputs,
- $W \in \mathbb{R}^{d' \times d}$ is a trainable weight matrix,
- note that the trainable bias vector of an MLP is omitted here for brevity.

Here is an example of a three-layer MLP,

$$\hat{y} = \sigma\left(\mathrm{ReLU}\left(\mathrm{ReLU}\left(XW^{(1)}\right)W^{(2)}\right)W^{(3)}\right).$$

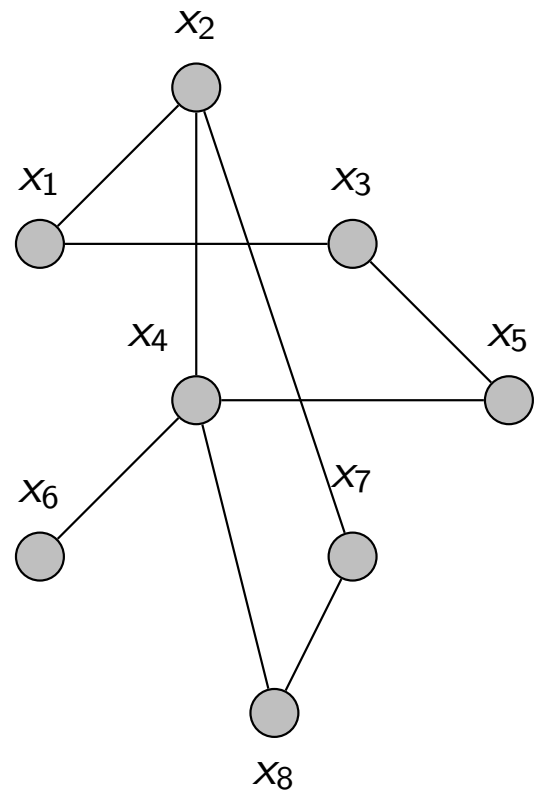
Graph Neural Networks

Graph Neural Networks (GNNs) are neural networks that take graph-structured data as input.

Graph Neural Networks

Graph Neural Networks (GNNs) are neural networks that take graph-structured data as input.

Today, we discuss a specific type of GNN, the Message Passing Neural Networks.



Graph Neural Networks

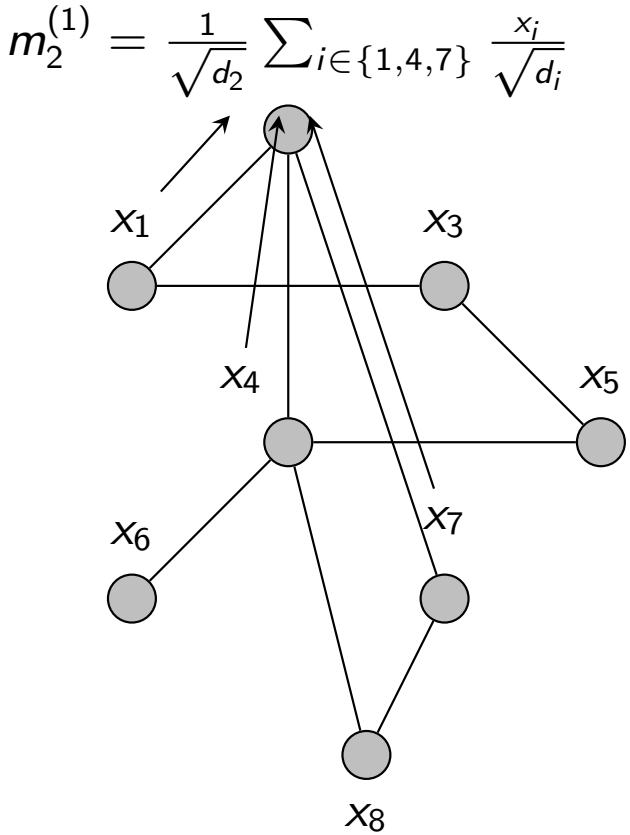
Graph Neural Networks (GNNs) are neural networks that take graph-structured data as input.

Today, we discuss a specific type of GNN, the Message Passing Neural Networks.

$$m_v^{(k)} = M^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right),$$

where

- $\mathcal{N}(v) = \{u : (u, v) \in E\}$ denotes the neighbourhood of a node v in $G = (V, E)$,
- $m_v^{(k)} \in \mathbb{R}^d$ denotes the representation of the messages received by neighbours of the node v,
- $M^{(k)}$ is a permutation-invariant function operating on sets.



E.g., the Graph Convolutional Network (GCN, Kipf and Welling, 2017)

$$\tilde{A}X,$$

where $\tilde{A} \in \mathbb{R}^{n \times n}$ and $X \in \mathbf{R}^{n \times d_0}$.

Graph Neural Networks

Graph Neural Networks (GNNs) are neural networks that take graph-structured data as input.

Today, we discuss a specific type of GNN, the Message Passing Neural Networks.

$$m_v^{(k)} = M^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right),$$
$$h_v^{(k)} = U^{(k)}\left(h_v^{(k-1)}, m_v^{(k)}\right).$$

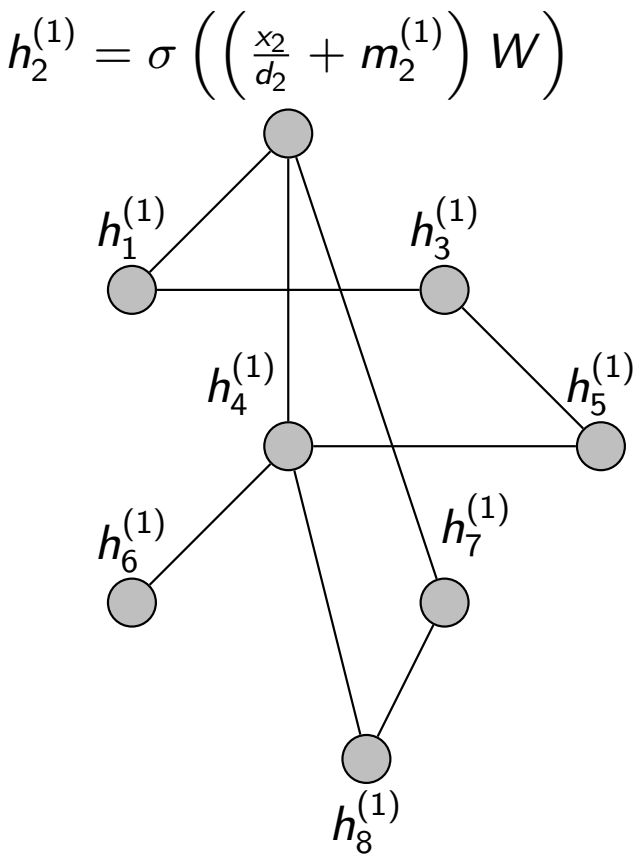
where

- $U^{(k)}$ is an often trainable update function combining $m_v^{(k)}$ and the representation of node v at the previous layer $h_v^{(k-1)}$.

E.g., the Graph Convolutional Network (GCN, Kipf and Welling, 2017)

$$H^{(1)} = \mathrm{ReLU}\left(\tilde{A}XW^{(1)}\right),$$

where $\tilde{A} \in \mathbb{R}^{n \times n}$, $X \in \mathbb{R}^{n \times d_0}$ and $W^{(1)} \in \mathbb{R}^{d_0 \times d_1}$.

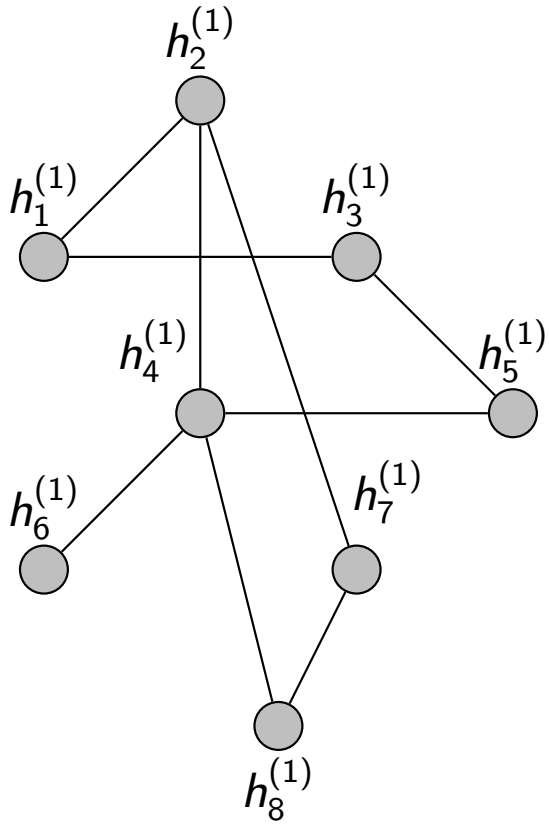


Graph Neural Networks

Graph Neural Networks (GNNs) are neural networks that take graph-structured data as input.

Today, we discuss a specific type of GNN, the Message Passing Neural Networks.

$$m_v^{(k)} = M^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right),$$
$$h_v^{(k)} = U^{(k)}\left(h_v^{(k-1)}, m_v^{(k)}\right).$$



E.g., the Graph Convolutional Network (GCN, Kipf and Welling, 2017)

$$H^{(1)} = \mathrm{ReLU}\left(\tilde{A}XW^{(1)}\right),$$

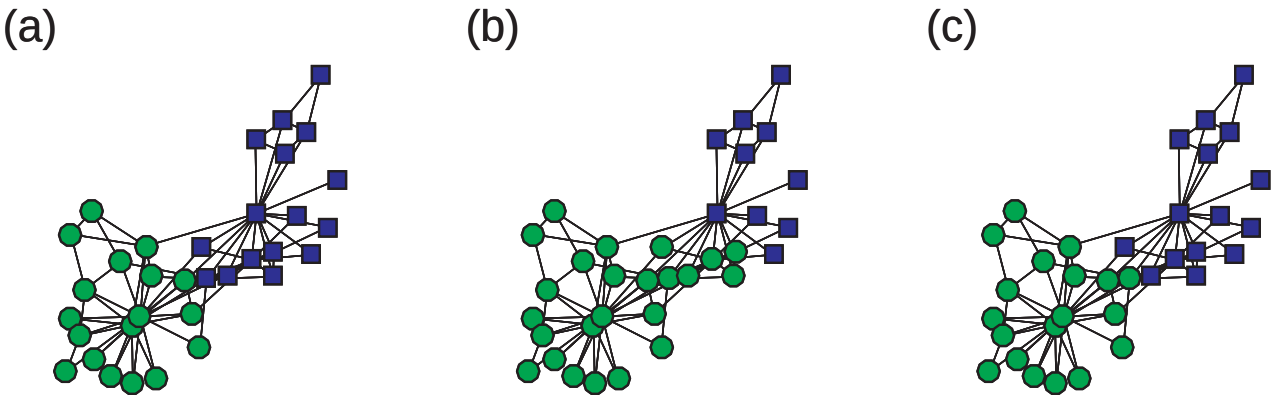
where $\tilde{A} \in \mathbb{R}^{n\times n}$, $X \in \mathbb{R}^{n\times d_0}$ and $W^{(1)} \in \mathbb{R}^{d_0\times d_1}$.

Iteratively performing the message-passing and update computations allows us to build 'deep' learning models, e.g., a 3-layer GCN

$$\hat{y} = \sigma\left(\tilde{A}\,\mathrm{ReLU}\left(\tilde{A}\,\mathrm{ReLU}\left(\tilde{A}XW^{(1)}\right)W^{(2)}\right)W^{(3)}\right).$$

GSOs in Graph Representation Learning

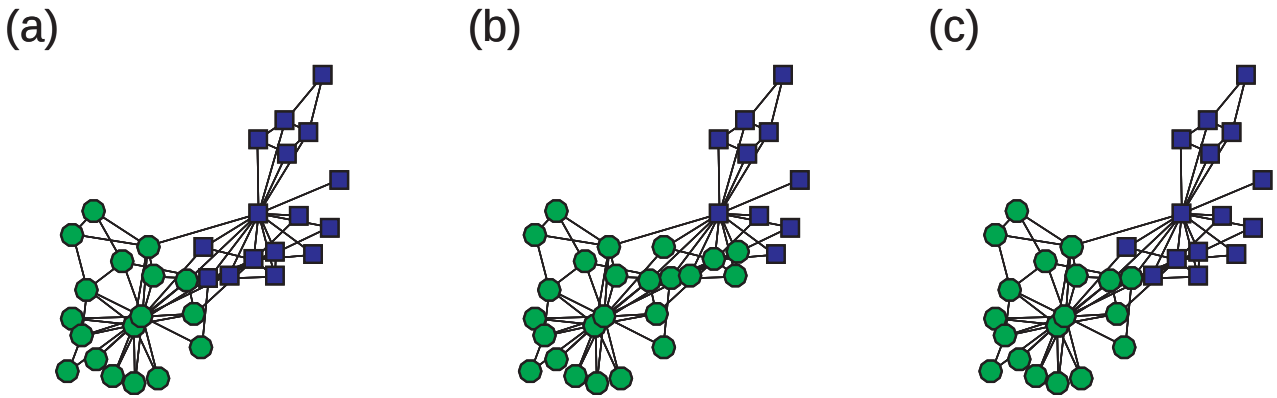
- Spectral clustering:



Spectral clustering of the karate network using A in (a), L in (b) and L_{rw} in (c) (Lutzeyer, 2020).

GSOs in Graph Representation Learning

- Spectral clustering:



Spectral clustering of the karate network using A in (a), L in (b) and L_{rw} in (c) (Lutzeyer, 2020).

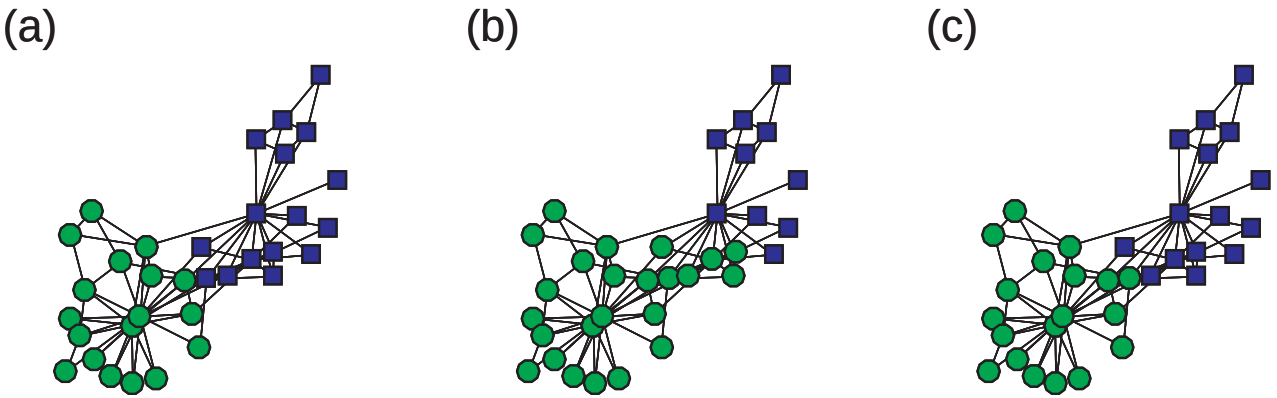
- Graph Neural Networks (GNNs), e.g., GCN (Kipf and Welling, 2017)

$$H^{(l+1)} = \sigma\left(D_1^{-\frac{1}{2}} A_1 D_1^{-\frac{1}{2}} H^{(l)} W^{(l)}\right). \tag{1}$$

The *sum-based aggregator* in the GIN (Xu et al., 2019) corresponds to the use of the adjacency matrix A.

GSOs in Graph Representation Learning

- Spectral clustering:



Spectral clustering of the karate network using A in (a), L in (b) and L_{rw} in (c) (Lutzeyer, 2020).

- Graph Neural Networks (GNNs), e.g., GCN (Kipf and Welling, 2017)

$$H^{(l+1)} = \sigma\left(D_1^{-\frac{1}{2}} A_1 D_1^{-\frac{1}{2}} H^{(l)} W^{(l)}\right). \tag{1}$$

The *sum-based aggregator* in the GIN (Xu et al., 2019) corresponds to the use of the adjacency matrix A.

In Message Passing Neural Networks, the choice of message passing function corresponds to a choice of GSO.

Academic and Industrial Success of GNNs

Empirical and Theoretical **Research**:

- expressivity analysis of GNNs (Xu et al., 2019; Geerts and Reutter, 2022);

Academic and Industrial Success of GNNs

Empirical and Theoretical **Research**:

- expressivity analysis of GNNs (Xu et al., 2019; Geerts and Reutter, 2022);
- robustness to adversarial attacks and noise (Günnemann, 2022; Zhou et al., 2020).

Academic and Industrial Success of GNNs

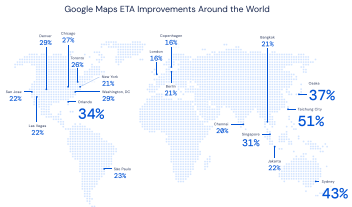
Empirical and Theoretical **Research**:

- expressivity analysis of GNNs (Xu et al., 2019; Geerts and Reutter, 2022);
- robustness to adversarial attacks and noise (Günnemann, 2022; Zhou et al., 2020).
- bottlenecks, e.g., oversmoothing and over-squashing (Alon and Yahav, 2020; Deac et al., 2022)

Academic and Industrial Success of GNNs

Empirical and Theoretical **Research**:

- expressivity analysis of GNNs (Xu et al., 2019; Geerts and Reutter, 2022);
- robustness to adversarial attacks and noise (Günnemann, 2022; Zhou et al., 2020).
- bottlenecks, e.g., oversmoothing and over-squashing (Alon and Yahav, 2020; Deac et al., 2022)



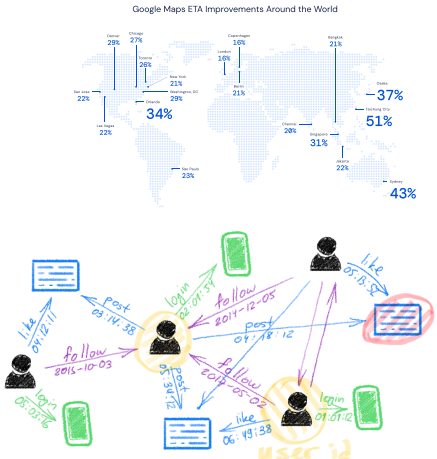
Successful **Applications** of GNNs:

- Google Maps (Lange and Perez, 2020);

Academic and Industrial Success of GNNs

Empirical and Theoretical **Research**:

- expressivity analysis of GNNs (Xu et al., 2019; Geerts and Reutter, 2022);
- robustness to adversarial attacks and noise (Günnemann, 2022; Zhou et al., 2020).
- bottlenecks, e.g., oversmoothing and over-squashing (Alon and Yahav, 2020; Deac et al., 2022)



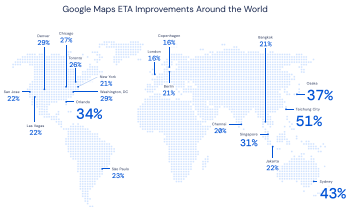
Successful **Applications** of GNNs:

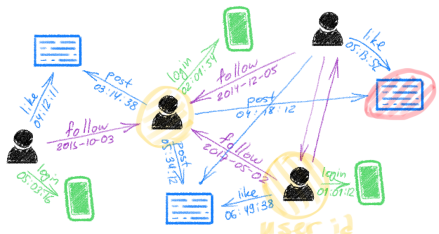
- Google Maps (Lange and Perez, 2020);
- Twitter (Bronstein, 2020);

Academic and Industrial Success of GNNs

Empirical and Theoretical **Research**:

- expressivity analysis of GNNs (Xu et al., 2019; Geerts and Reutter, 2022);
- robustness to adversarial attacks and noise (Günnemann, 2022; Zhou et al., 2020).
- bottlenecks, e.g., oversmoothing and over-squashing (Alon and Yahav, 2020; Deac et al., 2022)





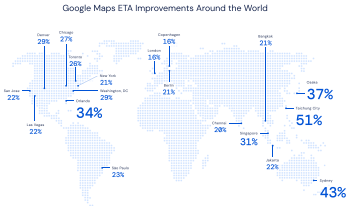
Successful **Applications** of GNNs:

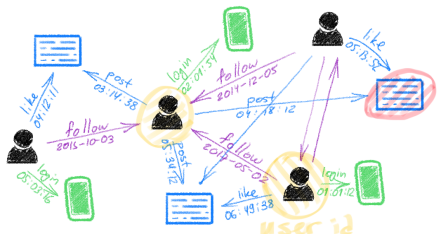
- Google Maps (Lange and Perez, 2020);
- Twitter (Bronstein, 2020);
- Amazon, Alibaba, Pinterest & Uber Eats (Virinchi et al., 2022; Wang et al., 2018; Ying et al., 2018; Jain et al., 2019);

Academic and Industrial Success of GNNs

Empirical and Theoretical **Research**:

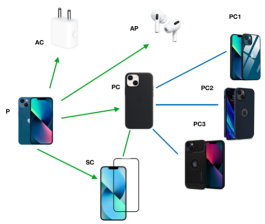
- expressivity analysis of GNNs (Xu et al., 2019; Geerts and Reutter, 2022);
- robustness to adversarial attacks and noise (Günnemann, 2022; Zhou et al., 2020).
- bottlenecks, e.g., oversmoothing and over-squashing (Alon and Yahav, 2020; Deac et al., 2022)





Successful **Applications** of GNNs:

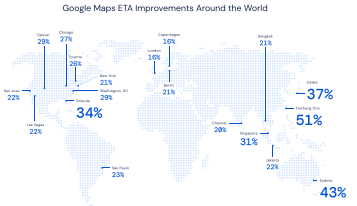
- Google Maps (Lange and Perez, 2020);
- Twitter (Bronstein, 2020);
- Amazon, Alibaba, Pinterest & Uber Eats (Virinchi et al., 2022; Wang et al., 2018; Ying et al., 2018; Jain et al., 2019);
- Discovery of two *new antibiotics* (Stokes et al., 2020; Liu et al., 2023);

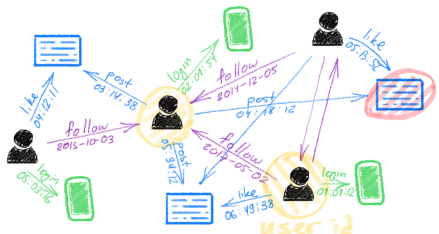


Academic and Industrial Success of GNNs

Empirical and Theoretical **Research**:

- expressivity analysis of GNNs (Xu et al., 2019; Geerts and Reutter, 2022);
- robustness to adversarial attacks and noise (Günnemann, 2022; Zhou et al., 2020).
- bottlenecks, e.g., oversmoothing and over-squashing (Alon and Yahav, 2020; Deac et al., 2022)





Successful **Applications** of GNNs:

- Google Maps (Lange and Perez, 2020);
- Twitter (Bronstein, 2020);
- Amazon, Alibaba, Pinterest & Uber Eats (Virinchi et al., 2022; Wang et al., 2018; Ying et al., 2018; Jain et al., 2019);
- Discovery of two *new antibiotics* (Stokes et al., 2020; Liu et al., 2023);
- LinkedIn (Borisyuk et al., 2024).

Different Approaches to Message Passing: Motivation

The message passing step is a defining component of GNNs.

Different Approaches to Message Passing: Motivation

The message passing step is a defining component of GNNs.

> *"any function of interest we want to compute over graphs can, in all likelihood, be expressed using pairwise message passing – just over a potentially modified graph [...]"*
>
> Petar Veličković (2022)

Different Approaches to Message Passing: Motivation

The message passing step is a defining component of GNNs.

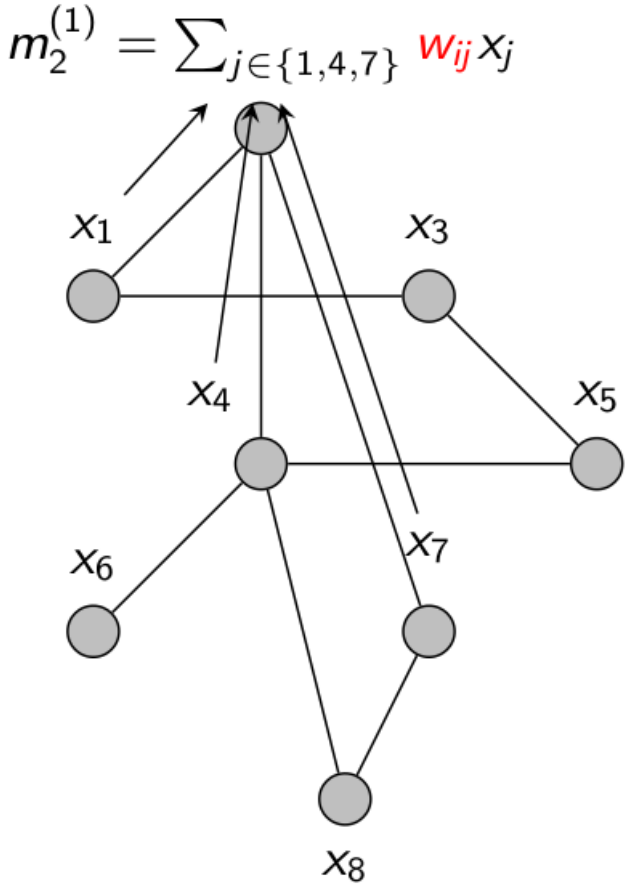
> *"any function of interest we want to compute over graphs can, in all likelihood, be expressed using pairwise message passing – just over a potentially modified graph [...]"*
>
> Petar Veličković (2022)

Topic for the Upcoming Segment of Today's Lecture

Categorise several existing GNNs by their message passing step.

Different Approaches to Message Passing

- Fixed Graph



Different Approaches to Message Passing

- Fixed Graph
  - GCN (Kipf and Welling, 2017)

    Message-Passing Operation: $D_1^{-\frac{1}{2}} A_1 D_1^{-\frac{1}{2}} X.$

Different Approaches to Message Passing

- Fixed Graph
  - GCN (Kipf and Welling, 2017)

    Message-Passing Operation: $D_1^{-\frac{1}{2}} A_1 D_1^{-\frac{1}{2}} X.$

  - GIN (Xu et al., 2019)

    Message-Passing Operation: $(A + \epsilon I) X.$

Different Approaches to Message Passing

- Fixed Graph
  - GCN (Kipf and Welling, 2017)

    Message-Passing Operation: $D_1^{-\frac{1}{2}} A_1 D_1^{-\frac{1}{2}} X.$

  - GIN (Xu et al., 2019)

    Message-Passing Operation: $(A + \epsilon I) X.$

  - PGSO-GNN (Dasoulas et al., 2021, ICLR)

    Message-Passing Operation:
    $\gamma(A, \mathcal{S}) = (m_1 D_a^{e_1} + m_2 D_a^{e_2} A_a D_a^{e_3} + m_3 I_n) X,$
    where $A_a = A + aI_n$, $D_a = \text{Diag}(A_a \mathbf{1}_n)$ and
    $(m_1, m_2, m_3, e_1, e_2, e_3, a)$ are scalar, *trainable* parameters.

$\mathcal{S} = (m_1, m_2, m_3, e_1, e_2, e_3, a)$	**Operator**	**Description**
$(0, 1, 0, 0, 0, 0, 0)$	A	Adjacency matrix and Summation Aggregation Operator of GNNs
$(1, -1, 0, 1, 0, 0, 0)$	$D - A$	Unnormalised Laplacian matrix L
$(1, 1, 0, 1, 0, 0, 0)$	$D + A$	Signless Laplacian matrix Q (Cvetkovic et al., 1997)
$(0, -1, 1, 0, -1, 0, 0)$	$I_n - D^{-1}A$	Random-walk Normalised Laplacian L_{rw}
$(0, -1, 1, 0, -\frac{1}{2}, -\frac{1}{2}, 0)$	$I_n - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$	Symmetric Normalised Laplacian L_{sym}
$(0, 1, 0, 0, -\frac{1}{2}, -\frac{1}{2}, 1)$	$D_1^{-\frac{1}{2}} A_1 D_1^{-\frac{1}{2}}$	Normalised Adjacency matrix of GCNs (Kipf and Welling, 2017)
$(0, 1, 0, 0, -1, 0, 0)$	$D^{-1}A$	Mean Aggregation Operator of GNNs (Xu et al., 2019)

PGSO in Graph Neural Networks

Notation:

- $\phi(A) : [0,1]^{n \times n} \to \mathbb{R}^{n \times n}$ denotes a non-parametrised function of A.
- $\mathcal{M}(\phi(A), X)$ denotes a GNN model.

PGSO in Graph Neural Networks

Notation:

- $\phi(A) : [0,1]^{n \times n} \to \mathbb{R}^{n \times n}$ denotes a non-parametrised function of A.
- $\mathcal{M}(\phi(A), X)$ denotes a GNN model.

Utilization of PGSO in GNNs

1. **GNN-PGSO**: $\mathcal{M}(\phi(A), X) \to \mathcal{M}'(\gamma(A, \mathcal{S}), X)$.
2. **GNN-mPGSO (multi-PGSO)**: $\mathcal{M}(\phi(A), X) \to \mathcal{M}''(\gamma^{[K]}(A, \mathcal{S}^{[K]}), X)$, where $\gamma^{[K]}(A, \mathcal{S}^{[K]}) = [\gamma(A, \mathcal{S}^1), \ldots, \gamma(A, \mathcal{S}^K)]$.

- Put simply, we **replace** the GSO used in a GNN model by $\gamma(A, \mathcal{S})$.

Convolutions and Message-Passing

- Examples of utilisation of GNN-PGSO models

Convolutions and Message-Passing

- Examples of utilisation of GNN-PGSO models

1. **GCN (Kipf & Welling, 2017)**: The propagation rule is

$$H^{(l+1)} = \sigma\big(D_1^{-\frac{1}{2}} A_1 D_1^{-\frac{1}{2}} H^{(l)} W^{(l)}\big),$$

where $W^{(l)}$ is a weight matrix and σ is a non-linear activation function. The GCN-PGSO and GCN-mPGSO models are defined, respectively, as

$$H^{(l+1)} = \sigma\big(\gamma(A, \mathcal{S}) H^{(l)} W^{(l)}\big) \text{ and } H^{(l+1)} = \sigma\big(\gamma(A, \mathcal{S}^l) H^{(l)} W^{(l)}\big).$$

Convolutions and Message-Passing

- Examples of utilisation of GNN-PGSO models

1. **GCN (Kipf & Welling, 2017)**: The propagation rule is

$$H^{(l+1)} = \sigma\left(D_1^{-\frac{1}{2}} A_1 D_1^{-\frac{1}{2}} H^{(l)} W^{(l)}\right),$$

where $W^{(l)}$ is a weight matrix and σ is a non-linear activation function. The GCN-PGSO and GCN-mPGSO models are defined, respectively, as

$$H^{(l+1)} = \sigma\left(\gamma(A, \mathcal{S}) H^{(l)} W^{(l)}\right) \text{ and } H^{(l+1)} = \sigma\left(\gamma(A, \mathcal{S}^l) H^{(l)} W^{(l)}\right).$$

2. **GIN (Xu et al., 2019)**: The propagation rule is

$$h_i^{(l+1)} = \sigma\left(h_i^{(l)} W^{(l)} + \sum_{j:v_j \in \mathcal{N}(v_i)} h_j^{(l)} W^{(l)}\right).$$

The GIN-PGSO model is defined as

$$h_i^{(l+1)} = \sigma\left(\left(m_1 \left(D_a\right)_i^{e_1} + m_3\right) h_i^{(l)} W^{(l)} + \sum_{j:v_j \in \mathcal{N}(v_i)} \epsilon_{ij} h_j^{(l)} W^{(l)}\right),$$

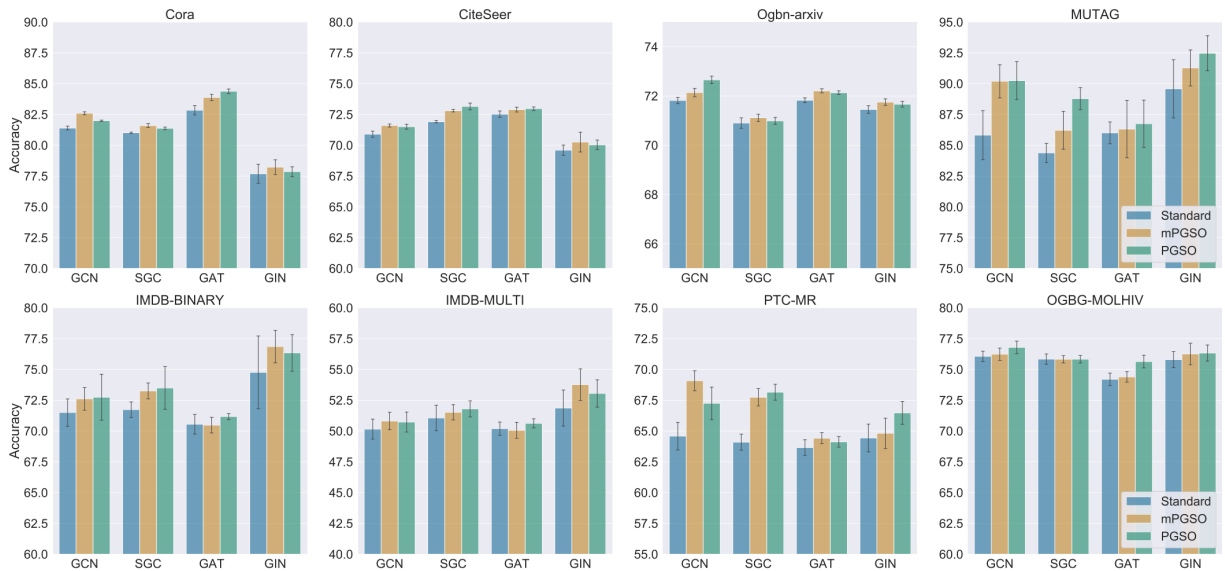
where ϵ_{ij} are edge weights defined as $\epsilon_{ij} = m_2 \left(D_a\right)_i^{e_2} \left(D_a\right)_j^{e_3}$.

Results on Real-world Datasets

- In **Simulation Studies** we independently verify theoretical results:
  - PGSO parameters replicate the GSO regularisation derived in Qin and Rohe (2013).

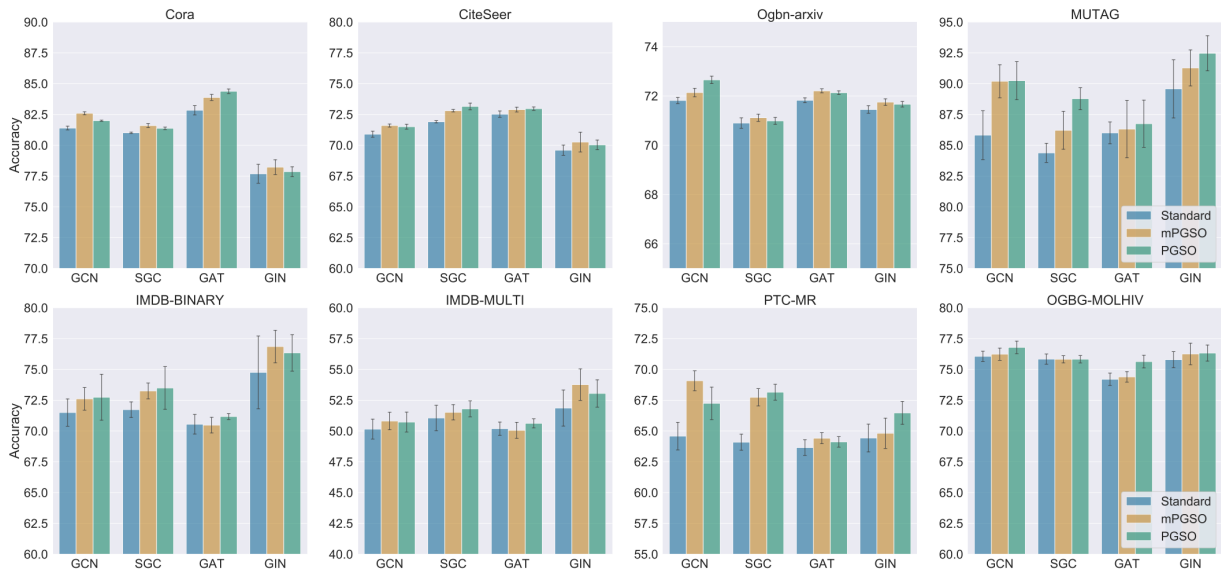
Results on Real-world Datasets

- In **Simulation Studies** we independently verify theoretical results:
  - PGSO parameters replicate the GSO regularisation derived in Qin and Rohe (2013).
- **Real-World Datasets**:
  - 3 node classification and 5 graph classification datasets.
  - 4 GNN architectures: GCN, SGC, GAT and GIN.
  - 3 GSO variants: **Standard**, **mPGSO** and **PGSO**.



Results on Real-world Datasets

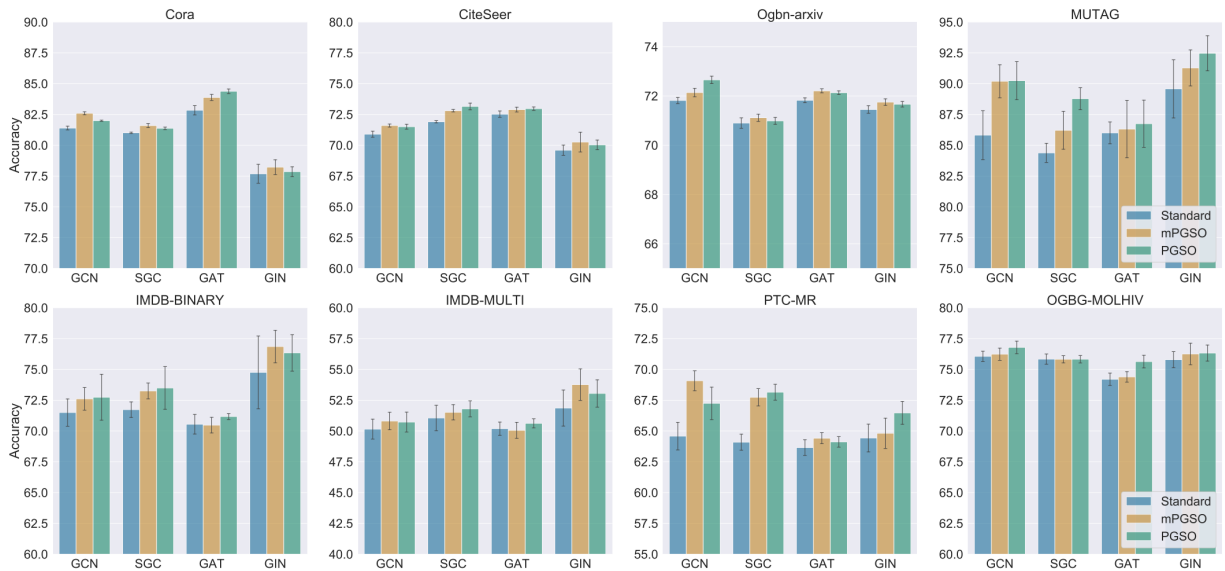
- In **Simulation Studies** we independently verify theoretical results:
  - PGSO parameters replicate the GSO regularisation derived in Qin and Rohe (2013).
- **Real-World Datasets**:
  - 3 node classification and 5 graph classification datasets.
  - 4 GNN architectures: GCN, SGC, GAT and GIN.
  - 3 GSO variants: **Standard**, **mPGSO** and **PGSO**.



- For all datasets and architectures, the incorporation of the PGSO and the mPGSO **enhances** the model performance.

Results on Real-world Datasets

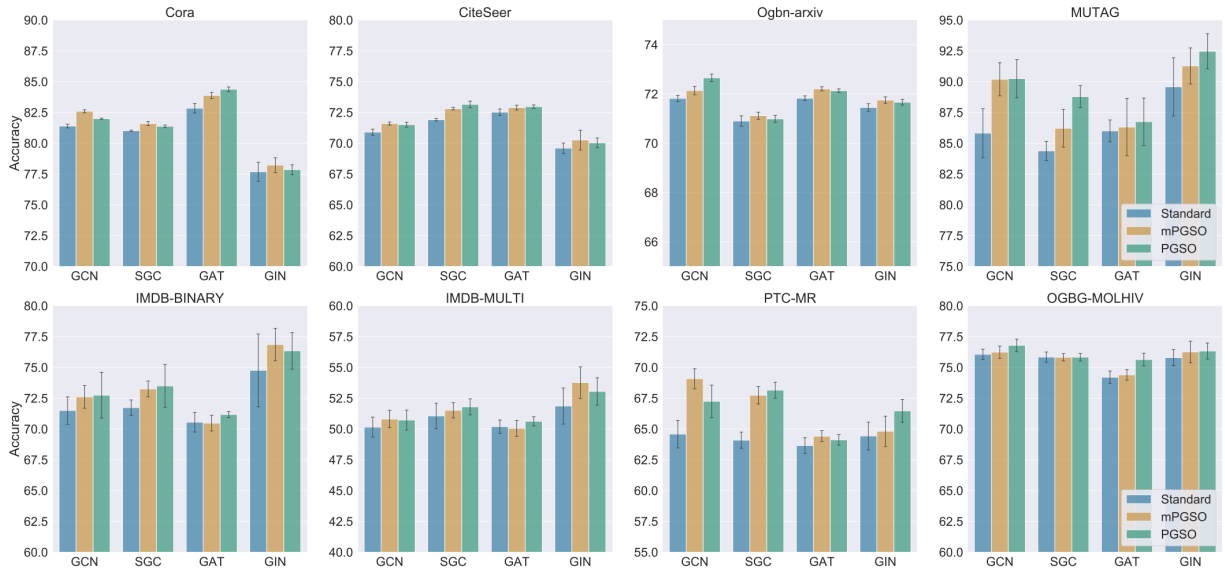
- In **Simulation Studies** we independently verify theoretical results:
  - PGSO parameters replicate the GSO regularisation derived in Qin and Rohe (2013).
- **Real-World Datasets**:
  - 3 node classification and 5 graph classification datasets.
  - 4 GNN architectures: GCN, SGC, GAT and GIN.
  - 3 GSO variants: **Standard**, **mPGSO** and **PGSO**.



- For all datasets and architectures, the incorporation of the PGSO and the mPGSO **enhances** the model performance.
- The impact of PGSO is **higher** in graph classification tasks.

Results on Real-world Datasets

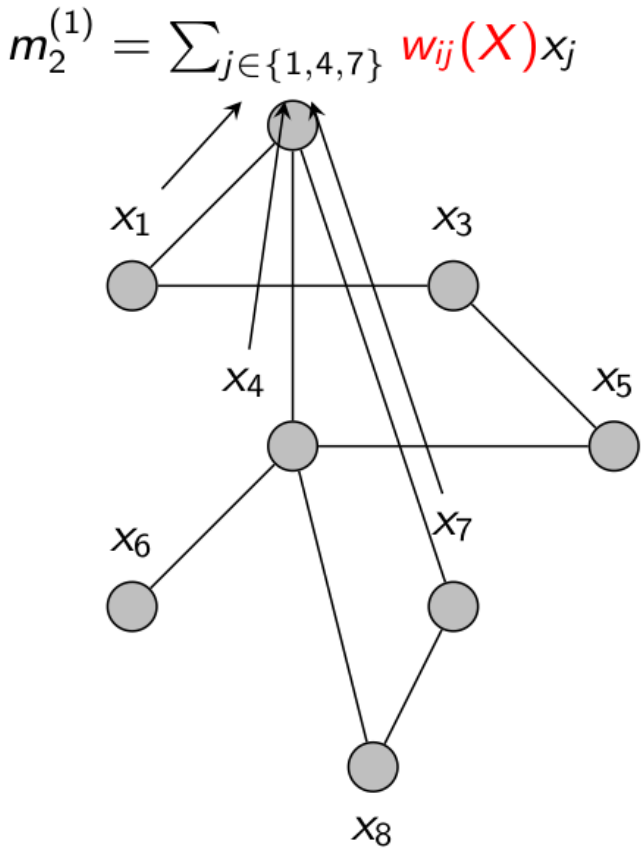
- In **Simulation Studies** we independently verify theoretical results:
  - PGSO parameters replicate the GSO regularisation derived in Qin and Rohe (2013).
- **Real-World Datasets**:
  - 3 node classification and 5 graph classification datasets.
  - 4 GNN architectures: GCN, SGC, GAT and GIN.
  - 3 GSO variants: **Standard**, **mPGSO** and **PGSO**.



- For all datasets and architectures, the incorporation of the PGSO and the mPGSO **enhances** the model performance.
- The impact of PGSO is **higher** in graph classification tasks.
- Our code is publicly available: https://github.com/gdasoulas/PGSO.

Different Approaches to Message Passing

- Fixed Graph
- Feature-Dependent Reweighting of Edges



Different Approaches to Message Passing

- Fixed Graph
- Feature-Dependent Reweighting of Edges
  - GAT (Veličković et al., 2018)
    Message-Passing Operation: $A_{att}X$,

$$\text{where } (A_{att})_{ij} = \begin{cases} 0, & \text{for } A_{ij} = 0; \\ \dfrac{\exp\left(\mathrm{LeakyReLU}\left(w_2^T \begin{bmatrix} W_1 h_i \\ W_1 h_j \end{bmatrix}\right)\right)}{\sum_{j \in \mathcal{N}(v_i)} \exp\left(\mathrm{LeakyReLU}\left(w_2^T \begin{bmatrix} W_1 h_i \\ W_1 h_j \end{bmatrix}\right)\right)}, & \text{for } A_{ij} \neq 0. \end{cases}$$

    Do you notice something in the arrangement of the weight matrices?

Different Approaches to Message Passing

- Fixed Graph
- Feature-Dependent Reweighting of Edges
  - GAT (Veličković et al., 2018)
    Message-Passing Operation: $A_{att}X$,

$$\text{where } (A_{att})_{ij} = \begin{cases} 0, & \text{for } A_{ij} = 0; \\ \dfrac{\exp\left(\mathrm{LeakyReLU}\left(w_2^T \begin{bmatrix} W_1 h_i \\ W_1 h_j \end{bmatrix}\right)\right)}{\sum_{j \in \mathcal{N}(v_i)} \exp\left(\mathrm{LeakyReLU}\left(w_2^T \begin{bmatrix} W_1 h_i \\ W_1 h_j \end{bmatrix}\right)\right)}, & \text{for } A_{ij} \neq 0. \end{cases}$$

  - GATv2 (Brody et al., 2022)
    Message-Passing Operation: $A_{attv2}X$,

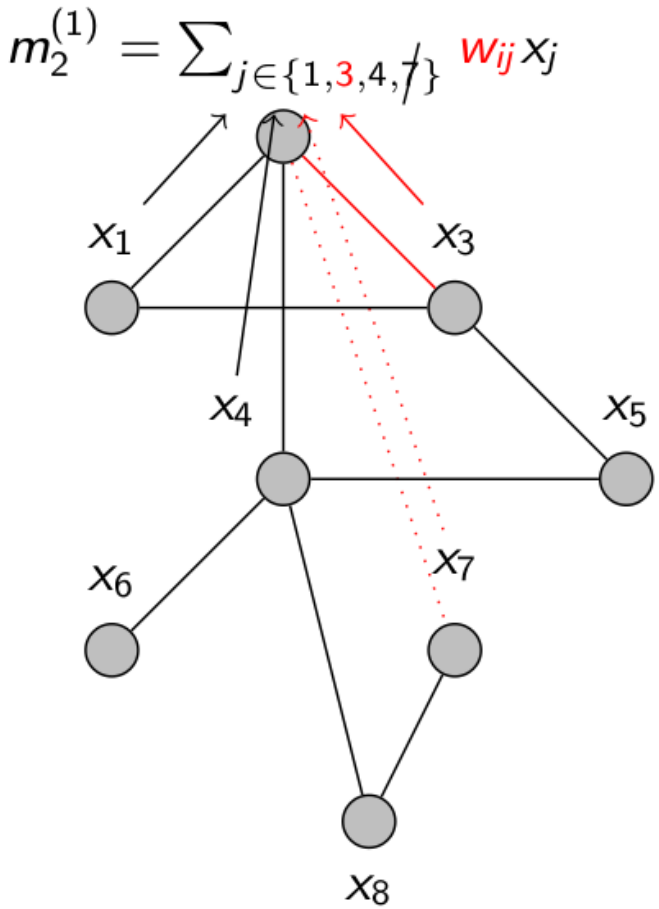
$$\text{where } (A_{attv2})_{ij} = \begin{cases} 0, & \text{for } A_{ij} = 0; \\ \dfrac{\exp\left(w_2^T \mathrm{LeakyReLU}\left(W_1 \begin{bmatrix} h_i \\ h_j \end{bmatrix}\right)\right)}{\sum_{j \in \mathcal{N}(v_i)} \exp\left(w_2^T \mathrm{LeakyReLU}\left(W_1 \begin{bmatrix} h_i \\ h_j \end{bmatrix}\right)\right)}, & \text{for } A_{ij} \neq 0. \end{cases}$$

Different Approaches to Message Passing

- Fixed Graph
- Feature-Dependent Reweighting of Edges
  - GAT (Veličković et al., 2018)
    Message-Passing Operation: $A_{att}X,$
  - GATv2 (Brody et al., 2022)
    Message-Passing Operation: $A_{attv2}X,$
  - GCN-k (Seddik et al., 2022, AISTATS)
    Message-Passing Operation: $\left(\epsilon D^{-\frac{1}{2}}AD^{-\frac{1}{2}} + (1-\epsilon)D_K^{-\frac{1}{2}}KD_K^{-\frac{1}{2}}\right)X,$
    where $(K)_{ij} = \begin{cases} 0, & \text{for } A_{ij} = 0; \\ x_i^T x_j, & \text{for } A_{ij} \neq 0. \end{cases}$

Different Approaches to Message Passing

- Fixed Graph
- Feature-Dependent Reweighting of Edges
- Adding and or Removing Edges



Different Approaches to Message Passing

- Fixed Graph
- Feature-Dependent Reweighting of Edges
- Adding and or Removing Edges
  - SDRF (Topping, Di Giovanni et al., 2022)
    Rewiring according to curvature metrics on graphs

Different Approaches to Message Passing

- Fixed Graph
- Feature-Dependent Reweighting of Edges
- Adding and or Removing Edges
  - SDRF (Topping, Di Giovanni et al., 2022)
    Rewiring according to curvature metrics on graphs
  - PPRGo (Bojchevski et al., 2020)
    Rewiring according to thresholded Personalised PageRank Scores

Different Approaches to Message Passing

- Fixed Graph
- Feature-Dependent Reweighting of Edges
- Adding and or Removing Edges
  - SDRF (Topping, Di Giovanni et al., 2022)
    Rewiring according to curvature metrics on graphs
  - PPRGo (Bojchevski et al., 2020)
    Rewiring according to thresholded Personalised PageRank Scores
  - CorePPR Ramos Vela et al. (2022, NeurIPS Workshop)
    Rewiring according to thresholded Personalised PageRank and CoreRank Scores

Different Approaches to Message Passing

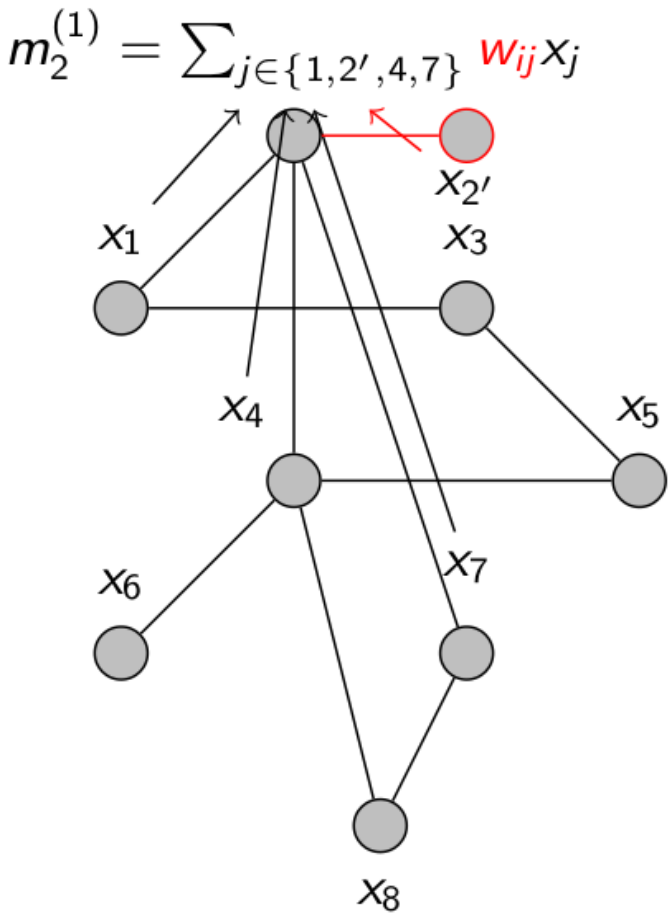
- Fixed Graph
- Feature-Dependent Reweighting of Edges
- Adding and or Removing Edges
  - SDRF (Topping, Di Giovanni et al., 2022)
    Rewiring according to curvature metrics on graphs
  - PPRGo (Bojchevski et al., 2020)
    Rewiring according to thresholded Personalised PageRank Scores
  - CorePPR Ramos Vela et al. (2022, NeurIPS Workshop)
    Rewiring according to thresholded Personalised PageRank and CoreRank Scores
  - Modularity-Aware (V)GAE (Salha-Galvan et al., 2022, Neural Networks)
    Add edges based on Louvain Clustering

Different Approaches to Message Passing

- Fixed Graph
- Feature-Dependent Reweighting of Edges
- Adding and or Removing Edges
  - SDRF (Topping, Di Giovanni et al., 2022)
    Rewiring according to curvature metrics on graphs
  - PPRGo (Bojchevski et al., 2020)
    Rewiring according to thresholded Personalised PageRank Scores
  - CorePPR Ramos Vela et al. (2022, NeurIPS Workshop)
    Rewiring according to thresholded Personalised PageRank and CoreRank Scores
  - Modularity-Aware (V)GAE (Salha-Galvan et al., 2022, Neural Networks)
    Add edges based on Louvain Clustering
  - Graph Transformers (Dwivedi and Bresson, 2021)
    Freely learn an adjacency matrix using the Transformer attention mechanism

Different Approaches to Message Passing

- Fixed Graph
- Feature-Dependent Reweighting of Edges
- Adding and or Removing Edges
- Explicitly Representing Substructures of Graphs



Different Approaches to Message Passing

- Fixed Graph
- Feature-Dependent Reweighting of Edges
- Adding and or Removing Edges
- Explicitly Representing Substructures of Graphs
    - Subgraph GNNs (Frasca et al., 2022)

Different Approaches to Message Passing

- Fixed Graph
- Feature-Dependent Reweighting of Edges
- Adding and or Removing Edges
- Explicitly Representing Substructures of Graphs
  - Subgraph GNNs (Frasca et al., 2022)
  - PathNNs (Michel et al., 2023, ICML)

Paths Collections

Idea

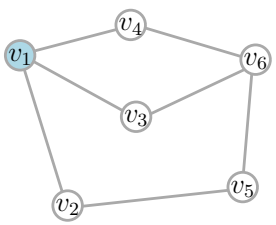
Explicitly representing paths (and nodes) instead of only nodes in GNNs should lead to more accurate and expressive models.

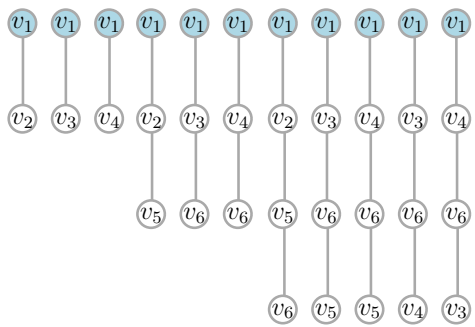
Paths Collections

Idea

Explicitly representing paths (and nodes) instead of only nodes in GNNs should lead to more accurate and expressive models.

We make use of three different collections of paths:





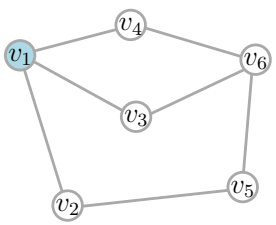
all paths ($\mathcal{AP}$)

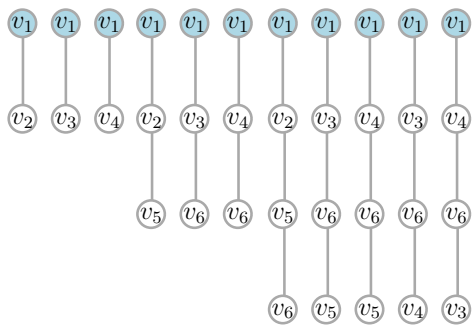
Paths Collections

Idea

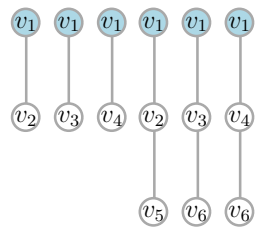
Explicitly representing paths (and nodes) instead of only nodes in GNNs should lead to more accurate and expressive models.

We make use of three different collections of paths:





all paths ($\mathcal{AP}$)



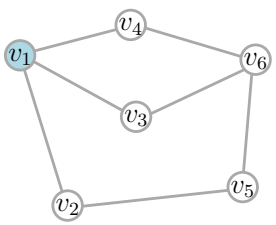
all shortest paths ($\mathcal{SP}^+$)

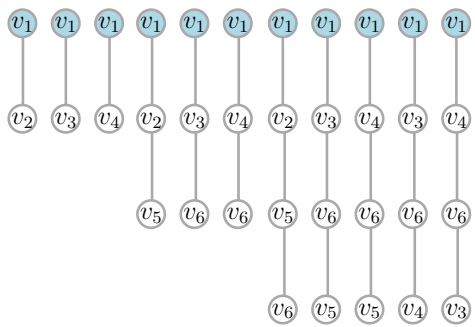
Paths Collections

Idea

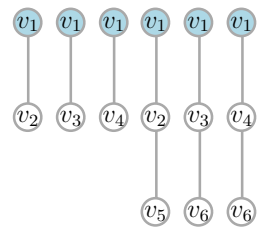
Explicitly representing paths (and nodes) instead of only nodes in GNNs should lead to more accurate and expressive models.

We make use of three different collections of paths:

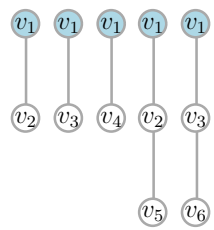




all paths ($\mathcal{AP}$)



all shortest paths ($\mathcal{SP}^+$)

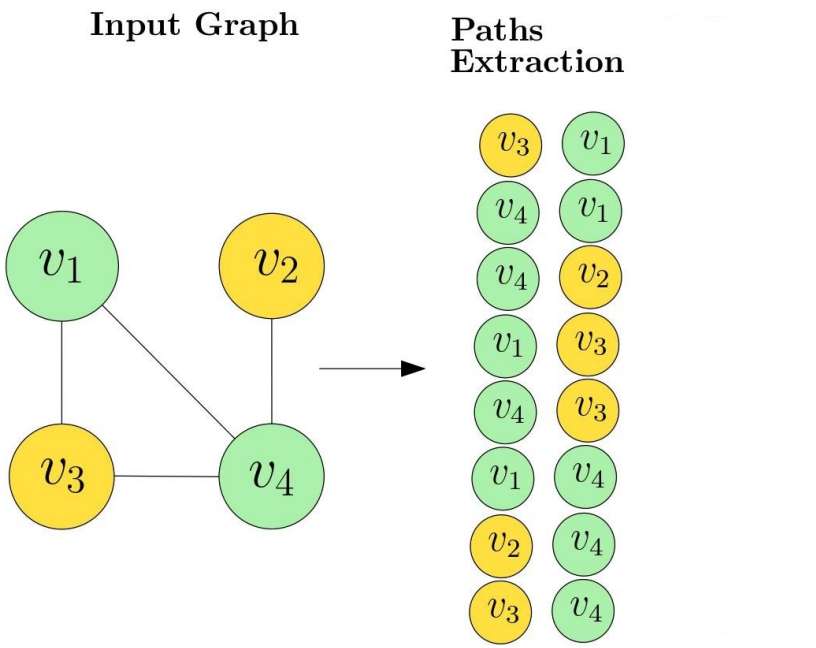


single shortest paths ($\mathcal{SP}$)

Path Neural Networks

1) Given a collection of paths in a graph,

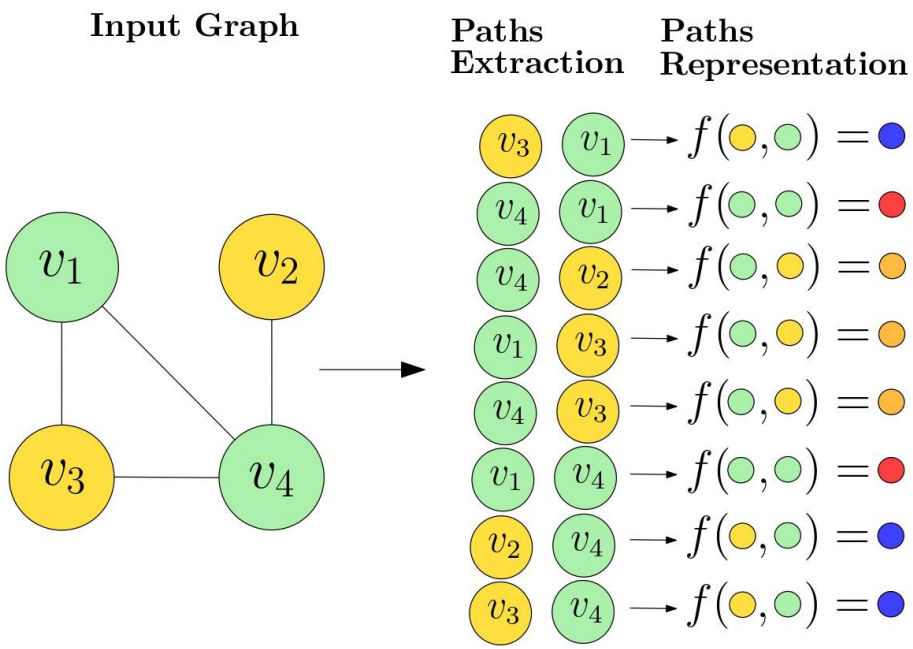
At layer ℓ of the model we process paths of length ℓ, e.g., $\ell = 2$.



Path Neural Networks

1) Given a collection of paths in a graph, we apply an LSTM to learn path representations.

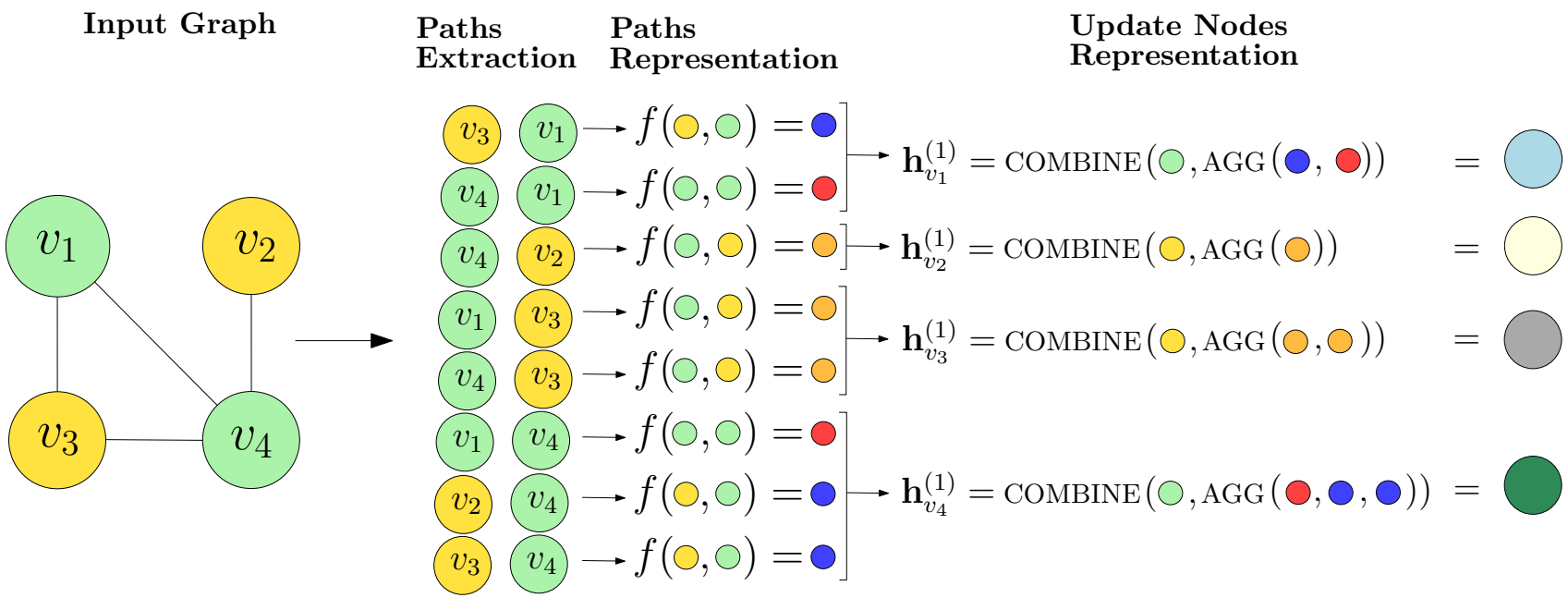
At layer ℓ of the model we process paths of length ℓ, e.g., $\ell = 2$.



Path Neural Networks

1) Given a collection of paths in a graph, we apply an LSTM to learn path representations.
2) We then aggregate the representations of all paths emanating from a node to form updated node representations.

At layer ℓ of the model we process paths of length ℓ, e.g., $\ell = 2$.



PathNNs – Theoretical Results

Standard GNNs can distinguish graphs at most as well as the Weisfeiler-Lehman (WL) test for graph isomorphism (Xu et al., 2019).

PathNNs – Theoretical Results

Standard GNNs can distinguish graphs at most as well as the Weisfeiler-Lehman (WL) test for graph isomorphism (Xu et al., 2019).

Theorem (Expressivity of PathNNs)

PathNN-$\mathcal{AP}$ and PathNN-$\mathcal{SP}^+$ are **strictly more powerful** than the WL test.

PathNNs – Theoretical Results

Standard GNNs can distinguish graphs at most as well as the Weisfeiler-Lehman (WL) test for graph isomorphism (Xu et al., 2019).

Theorem (Expressivity of PathNNs)

PathNN-$\mathcal{AP}$ and PathNN-$\mathcal{SP}^+$ are **strictly more powerful** than the WL test.

PathNN-$\mathcal{SP}$ also **distinguishes any two graphs distinguished by the WL test**. But in the presence of multiple shortest paths between two vertices a particular sampling of these shortest paths could lead to the PathNN-$\mathcal{SP}$ distinguishing even structurally equivalent nodes.

Results on Real-World Datasets

- PathNNs demonstrate convincing performance on **Synthetic Datasets** designed to benchmark the expressivity of GNNs.

Results on Real-World Datasets

- PathNNs demonstrate convincing performance on **Synthetic Datasets** designed to benchmark the expressivity of GNNs.
- **6 Real-World Graph Classification Datasets**

	DD	PROTEINS	NCI1	ENZYMES	IMDB-B	IMDB-M
GIN	75.3 ± 2.9	73.3 ± 4.0	80.0 ± 1.4	59.6 ± 4.5	71.2 ± 3.9	48.5 ± 3.3
GraphSAGE	72.9 ± 2.0	73.0 ± 4.5	76.0 ± 1.8	58.2 ± 6.0	68.8 ± 4.5	47.6 ± 3.5
GAT	73.9 ± 3.4	70.9 ± 2.7	77.3 ± 2.5	49.5 ± 8.9	69.2 ± 4.8	48.2 ± 4.9
SPN ($K = 1$)	72.7 ± 2.6	71.0 ± 3.7	80.0 ± 1.5	67.5 ± 5.5	NA	NA
SPN ($K = 5$)	77.4 ± 3.8	74.2 ± 2.7	78.6 ± 3.8	69.4 ± 6.2	NA	NA
PathNet ($N = 10, K = 2$)	OOM	70.5 ± 3.9	64.1 ± 2.3	69.3 ± 5.4	70.4 ± 3.8	49.1 ± 3.6
Nested GNN	**77.8** ± 3.9	74.2 ± 3.7	NA	31.2 ± 6.7	NA	NA
PathNN-$\mathcal{P}$ ($K = 1$)	76.9 ± 3.7	**75.2** ± 3.9	77.5 ± 1.6	**73.0** ± 5.2	**72.6** ± 3.3	**50.8** ± 4.5
PathNN-$\mathcal{SP}$ ($K = 2$)	75.3 ± 2.7	73.1 ± 3.1	82.0 ± 1.6	71.6 ± 6.4	70.8 ± 3.5	50.0 ± 4.1
PathNN-$\mathcal{SP}$ ($K = 3$)	77.0 ± 3.1	72.2 ± 2.7	82.2 ± 1.7	69.2 ± 4.7	-	-
PathNN-$\mathcal{SP}^+$ ($K = 2$)	74.7 ± 3.0	73.1 ± 3.7	81.0 ± 1.4	72.5 ± 5.3	70.5 ± 3.4	50.7 ± 4.5
PathNN-$\mathcal{SP}^+$ ($K = 3$)	76.5 ± 4.6	73.2 ± 3.3	**82.3** ± 1.9	70.4 ± 3.1	-	-
PathNN-$\mathcal{AP}$ ($K = 2$)	75.0 ± 4.4	73.1 ± 4.9	81.3 ± 1.8	71.8 ± 4.8	71.7 ± 3.6	49.8 ± 4.2
PathNN-$\mathcal{AP}$ ($K = 3$)	OOM	73.1 ± 4.0	82.3 ± 1.7	69.0 ± 5.3	OOM	OOM

- Our code is publicly available: https://github.com/gasmichel/PathNNs_expressive

Different Approaches to Message Passing

- Fixed Graph
- Feature-Dependent Reweighting of Edges
- Adding and or Removing Edges
- Explicitly Representing Substructures of Graphs
- Some GNNs are difficult categorise
    - GOAT (Chatzianastasis, et al., 2023, AAAI)

Different Approaches to Message Passing

- Fixed Graph
- Feature-Dependent Reweighting of Edges
- Adding and or Removing Edges
- Explicitly Representing Substructures of Graphs
- Some GNNs are difficult categorise
  - GOAT (Chatzianastasis, et al., 2023, AAAI)
    1) A self-attention mechanism is used to obtain a ranking of nodes in neighbourhoods.

Different Approaches to Message Passing

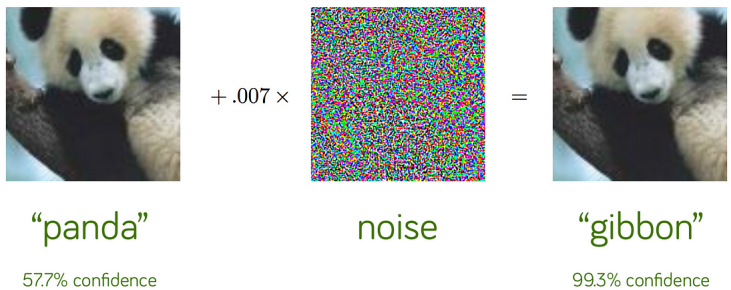
- Fixed Graph
- Feature-Dependent Reweighting of Edges
- Adding and or Removing Edges
- Explicitly Representing Substructures of Graphs
- Some GNNs are difficult categorise
  - GOAT (Chatzianastasis, et al., 2023, AAAI)
    1) A self-attention mechanism is used to obtain a ranking of nodes in neighbourhoods.
    2) An LSTM processed the ordered neighbourhoods to produce updated node representation.

1) Bounding the Expected Robustness of Graph Neural Networks Subject to Node Feature Attacks

Abbahaddou*, et al. (2024, ICLR)

(Graph) Adversarial Attacks

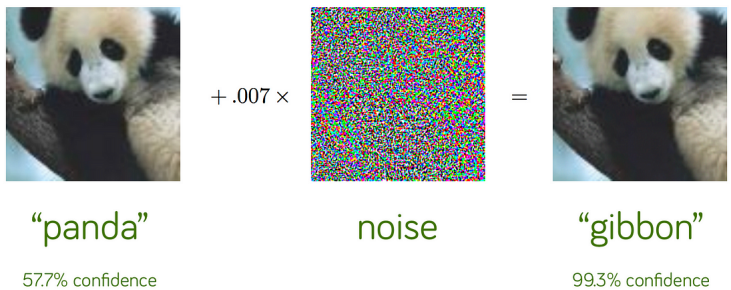
Goal: Adversarial attacks apply a *small* change to the input to achieve a *large* change in the output of our model.



(Goodfellow et al., 2015)

(Graph) Adversarial Attacks

Goal: Adversarial attacks apply a *small* change to the input to achieve a *large* change in the output of our model.

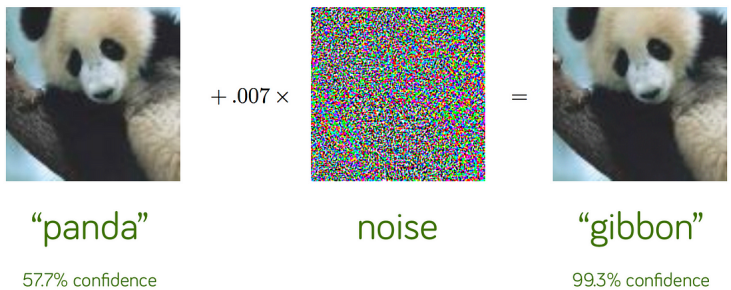


(Goodfellow et al., 2015)

To quantify the robustness of a function processing graph structured data, i.e., $f : (\mathcal{G}, \mathcal{X}) \to \mathcal{Y}$ we need:

(Graph) Adversarial Attacks

Goal: Adversarial attacks apply a *small* change to the input to achieve a *large* change in the output of our model.



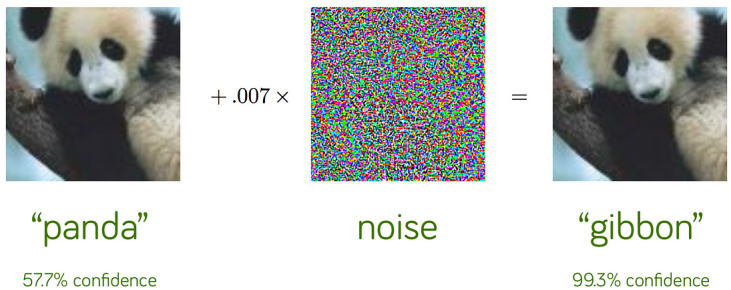
(Goodfellow et al., 2015)

To quantify the robustness of a function processing graph structured data, i.e., $f : (\mathcal{G}, \mathcal{X}) \to \mathcal{Y}$ we need:

- a distance on the input space
  $d_2^{\alpha,\beta}([G, X], [\tilde{G}, \tilde{X}]) = \min_{P \in \Pi} \left(\alpha \|A - P\tilde{A}P^T\|_2 + \beta \|X - P\tilde{X}\|_2 \right),$

(Graph) Adversarial Attacks

Goal: Adversarial attacks apply a *small* change to the input to achieve a *large* change in the output of our model.



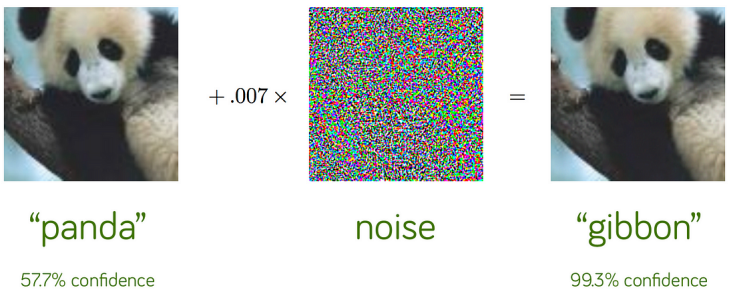
(Goodfellow et al., 2015)

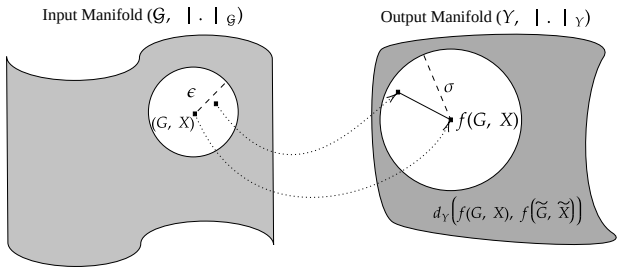
To quantify the robustness of a function processing graph structured data, i.e., $f : (\mathcal{G}, \mathcal{X}) \to \mathcal{Y}$ we need:

- a distance on the input space
  $d_2^{\alpha,\beta}([G, X], [\tilde{G}, \tilde{X}]) = \min_{P \in \Pi} \left(\alpha \|A - P\tilde{A}P^T\|_2 + \beta \|X - P\tilde{X}\|_2 \right),$
- and a distance on the output space
  $d_1(f(\tilde{G}, \tilde{X}), f(G, X)) = \|f(\tilde{G}, \tilde{X}) - f(G, X)\|_1.$

(Graph) Adversarial Attacks

Goal: Adversarial attacks apply a *small* change to the input to achieve a *large* change in the output of our model.





(Goodfellow et al., 2015)

To quantify the robustness of a function processing graph structured data, i.e., $f : (\mathcal{G}, \mathcal{X}) \to \mathcal{Y}$ we need:

- a distance on the input space
  $d_2^{\alpha,\beta}([G, X], [\tilde{G}, \tilde{X}]) = \min_{P \in \Pi} \left(\alpha \|A - P\tilde{A}P^T\|_2 + \beta \|X - P\tilde{X}\|_2 \right),$
- and a distance on the output space
  $d_1(f(\tilde{G}, \tilde{X}), f(G, X)) = \|f(\tilde{G}, \tilde{X}) - f(G, X)\|_1.$

(Graph) Adversarial Attacks

Goal: Adversarial attacks apply a *small* change to the input to achieve a *large* change in the output of our model.

To quantify the robustness of a function processing graph structured data, i.e., $f : (\mathcal{G}, \mathcal{X}) \to \mathcal{Y}$ we need:

- a distance on the input space
  $d_2^{\alpha,\beta}([G, X], [\tilde{G}, \tilde{X}]) = \min_{P \in \Pi} \left(\alpha \|A - P\tilde{A}P^T\|_2 + \beta \|X - P\tilde{X}\|_2 \right),$
- and a distance on the output space
  $d_1(f(\tilde{G}, \tilde{X}), f(G, X)) = \|f(\tilde{G}, \tilde{X}) - f(G, X)\|_1.$

Expected Adversarial Robustness

Let the *expected vulnerability* of a graph function f be defined as
$Adv_\epsilon^{\alpha,\beta}[f] = \mathbb{P}_{(G,X)\sim\mathcal{D}_{\mathcal{G},\mathcal{X}}}[(\tilde{G}, \tilde{X}) \in B^{\alpha,\beta}(G, X, \epsilon) : d_{\mathcal{Y}}(f(\tilde{G}, \tilde{X}), f(G, X)) > \sigma]$,
with $B^{\alpha,\beta}(G, X, \epsilon) = \{(\tilde{G}, \tilde{X}) : d^{\alpha,\beta}([G, X], [\tilde{G}, \tilde{X}]) < \epsilon\}$ for any budget $\epsilon \geq 0$.

(Graph) Adversarial Attacks

Goal: Adversarial attacks apply a *small* change to the input to achieve a *large* change in the output of our model.

To quantify the robustness of a function processing graph structured data, i.e., $f : (\mathcal{G}, \mathcal{X}) \to \mathcal{Y}$ we need:

- a distance on the input space
  $d_2^{\alpha,\beta}([G, X], [\tilde{G}, \tilde{X}]) = \min_{P \in \Pi} \left(\alpha \|A - P\tilde{A}P^T\|_2 + \beta \|X - P\tilde{X}\|_2 \right),$
- and a distance on the output space
  $d_1(f(\tilde{G}, \tilde{X}), f(G, X)) = \|f(\tilde{G}, \tilde{X}) - f(G, X)\|_1.$

Expected Adversarial Robustness

Let the *expected vulnerability* of a graph function f be defined as
$Adv_\epsilon^{\alpha,\beta}[f] = \mathbb{P}_{(G,X)\sim\mathcal{D}_{\mathcal{G},\mathcal{X}}}[(\tilde{G}, \tilde{X}) \in B^{\alpha,\beta}(G, X, \epsilon) : d_{\mathcal{Y}}(f(\tilde{G}, \tilde{X}), f(G, X)) > \sigma]$,
with $B^{\alpha,\beta}(G, X, \epsilon) = \{(\tilde{G}, \tilde{X}) : d^{\alpha,\beta}([G, X], [\tilde{G}, \tilde{X}]) < \epsilon\}$ for any budget $\epsilon \geq 0$.

Then, a graph function $f : (\mathcal{G}, \mathcal{X}) \to \mathcal{Y}$ is $((d^{\alpha,\beta}, \epsilon), (d_{\mathcal{Y}}, \gamma))$–*robust* if its vulnerability $Adv_\epsilon^{\alpha,\beta}[f]$ can be upper-bounded by γ, i.e., $Adv_\epsilon^{\alpha,\beta}[f] \leq \gamma$.

Problem Set-Up & Theoretical Results

Recall, Graph Neural Networks (GNNs) take both a graph A and node features X as input.

Problem Set-Up & Theoretical Results

Recall, Graph Neural Networks (GNNs) take both a graph A and node features X as input.

Problem: Most defense approaches for GNNs defend structural attacks altering A. There exists very little work on how to defend against attacks on the node features X.

Problem Set-Up & Theoretical Results

Recall, Graph Neural Networks (GNNs) take both a graph A and node features X as input.

Problem: Most defense approaches for GNNs defend structural attacks altering A. There exists very little work on how to defend against attacks on the node features X.

Upper Bound on GCN Vulnerability

We consider node-feature attacks on the input graph (A, X), with a budget ϵ and L-layer GCNs with weight matrices $W^{(i)}$ for $i \in \{1, \ldots, L\}$.
Then, the vulnerability of GCNs is upper bounded by

$$\gamma = \prod_{i=1}^{L} \|W^{(i)}\|_1 \frac{\epsilon \sum_{u \in \mathcal{V}} \hat{w}_u}{\sigma},$$

with $\hat{w}_u$ denoting the sum of normalized walks of length $(L-1)$ starting from node u.

Problem Set-Up & Theoretical Results

Recall, Graph Neural Networks (GNNs) take both a graph A and node features X as input.

Problem: Most defense approaches for GNNs defend structural attacks altering A. There exists very little work on how to defend against attacks on the node features X.

Upper Bound on GCN Vulnerability

We consider node-feature attacks on the input graph (A, X), with a budget ϵ and L-layer GCNs with weight matrices $W^{(i)}$ for $i \in \{1, \ldots, L\}$.
Then, the vulnerability of GCNs is upper bounded by

$$\gamma = \prod_{i=1}^{L} \|W^{(i)}\|_1 \frac{\epsilon \sum_{u \in \mathcal{V}} \hat{w}_u}{\sigma},$$

with $\hat{w}_u$ denoting the sum of normalized walks of length $(L-1)$ starting from node u.

Insight: Our upper bound on the vulnerability of a GCN is **smaller for small** $\prod_{i=1}^{L} \|W^{(i)}\|_1$ yielding a **more robust GCN.**

Methodology

Fact: Orthonormal matrices have norm 1.

$\Rightarrow$ According to our bound a GNN with orthonormal weight matrices should be more robust.

Methodology

Fact: Orthonormal matrices have norm 1.

$\Rightarrow$ According to our bound a GNN with orthonormal weight matrices should be more robust.

Björk Orthonormalisation Algorithm

Given a weight matrix W we iteratively alter it to approximate the closest orthonormal matrix $\hat{W}$. When $\hat{W}_0 = W$, we recursively compute

$$\hat{W}_{k+1} = \hat{W}_k \left(I + \tfrac{1}{2} \left(I - \hat{W}_k^T \hat{W}_k \right) + \ldots + (-1)^p \binom{-1/2}{p} \left(I - \hat{W}_k^T \hat{W}_k \right)^p \right).$$

Methodology

Fact: Orthonormal matrices have norm 1.

$\Rightarrow$ According to our bound a GNN with orthonormal weight matrices should be more robust.

Björk Orthonormalisation Algorithm

Given a weight matrix W we iteratively alter it to approximate the closest orthonormal matrix $\hat{W}$. When $\hat{W}_0 = W$, we recursively compute

$$\hat{W}_{k+1} = \hat{W}_k \left(I + \tfrac{1}{2} \left(I - \hat{W}_k^T \hat{W}_k \right) + \ldots + (-1)^p \binom{-1/2}{p} \left(I - \hat{W}_k^T \hat{W}_k \right)^p \right).$$

Proposed Solution: In our *GCORN* model we propose the inclusion of several Björk Orthonormalisation iterations in each forward pass during the training of a GCN, **yielding weight matrices that approach orthonormality and thereby a more robust GNN**.

Results

Table: Node classification accuracy ($\pm$ standard deviation) for feature-based attacks.

Attack	Dataset	GCN	GCN-k	AirGNN	RGCN	ParsevalR	GCORN
Random ($\psi = 0.5$)	Cora	68.4 $\pm$ 1.9	69.2 $\pm$ 2.6	73.5 $\pm$ 1.9	71.6 $\pm$ 0.3	72.9 $\pm$ 0.9	**77.1 $\pm$ 1.8**
	CiteSeer	57.8 $\pm$ 1.5	62.3 $\pm$ 1.2	64.6 $\pm$ 1.6	63.7 $\pm$ 0.6	65.1 $\pm$ 0.8	**67.8 $\pm$ 1.4**
	PubMed	68.3 $\pm$ 1.2	71.2 $\pm$ 1.1	70.9 $\pm$ 1.3	71.4 $\pm$ 0.5	71.8 $\pm$ 0.8	**73.1 $\pm$ 1.1**
	CS	85.3 $\pm$ 1.1	86.7 $\pm$ 1.1	87.5 $\pm$ 1.6	88.2 $\pm$ 0.9	87.6 $\pm$ 0.6	**89.8 $\pm$ 1.2**
	OGBN-Arxiv	68.2 $\pm$ 1.5	52.8 $\pm$ 0.5	66.5 $\pm$ 1.3	63.8 $\pm$ 1.9	68.3 $\pm$ 1.9	**69.1 $\pm$ 1.8**
Random ($\psi = 1.0$)	Cora	41.7 $\pm$ 2.1	46.3 $\pm$ 2.8	53.7 $\pm$ 2.2	52.8 $\pm$ 1.6	55.3 $\pm$ 1.2	**57.6 $\pm$ 1.9**
	CiteSeer	38.2 $\pm$ 1.3	45.3 $\pm$ 1.4	49.8 $\pm$ 2.1	43.7 $\pm$ 2.2	51.2 $\pm$ 1.2	**57.3 $\pm$ 1.7**
	PubMed	60.1 $\pm$ 1.7	62.3 $\pm$ 1.3	62.4 $\pm$ 1.2	61.9 $\pm$ 1.2	61.3 $\pm$ 1.7	**65.8 $\pm$ 1.4**
	CS	69.9 $\pm$ 1.3	73.2 $\pm$ 0.9	76.7 $\pm$ 2.8	76.2 $\pm$ 1.4	78.7 $\pm$ 1.2	**81.3 $\pm$ 1.6**
	OGBN-Arxiv	66.4 $\pm$ 1.9	46.6 $\pm$ 0.6	62.7 $\pm$ 1.6	63.0 $\pm$ 2.4	66.1 $\pm$ 0.7	**67.3 $\pm$ 2.1**
PGD	Cora	54.1 $\pm$ 2.4	58.3 $\pm$ 1.6	68.2 $\pm$ 1.8	62.5 $\pm$ 1.2	68.6 $\pm$ 1.7	**71.1 $\pm$ 1.4**
	CiteSeer	52.3 $\pm$ 1.1	59.6 $\pm$ 1.6	59.3 $\pm$ 2.1	61.9 $\pm$ 1.1	62.1 $\pm$ 1.5	**65.6 $\pm$ 1.4**
	PubMed	66.1 $\pm$ 2.1	67.3 $\pm$ 1.3	70.8 $\pm$ 1.7	69.5 $\pm$ 0.9	68.9 $\pm$ 2.1	**72.3 $\pm$ 1.3**
	CS	71.3 $\pm$ 1.1	74.1 $\pm$ 0.8	76.3 $\pm$ 2.1	76.6 $\pm$ 1.2	77.3 $\pm$ 0.6	**79.6 $\pm$ 1.2**
	OGBN-Arxiv	67.5 $\pm$ 0.9	49.9 $\pm$ 0.7	55.7 $\pm$ 0.9	63.6 $\pm$ 0.7	67.6 $\pm$ 1.2	**68.1 $\pm$ 1.1**
Nettack	Cora	60.9 $\pm$ 2.5	64.2 $\pm$ 5.2	66.7 $\pm$ 3.8	63.4 $\pm$ 3.8	67.5 $\pm$ 2.5	**68.3 $\pm$ 1.4**
	CiteSeer	55.8 $\pm$ 1.4	71.7 $\pm$ 1.4	67.5 $\pm$ 2.5	70.8 $\pm$ 3.8	69.2 $\pm$ 3.8	**77.5 $\pm$ 2.5**
	PubMed	60.0 $\pm$ 2.5	65.8 $\pm$ 2.9	69.2 $\pm$ 1.4	**71.7 $\pm$ 3.8**	68.3 $\pm$ 1.4	70.8 $\pm$ 1.4
	CS	55.8 $\pm$ 1.4	71.6 $\pm$ 1.4	76.7 $\pm$ 1.4	71.7 $\pm$ 2.9	75.8 $\pm$ 2.8	**78.3 $\pm$ 1.4**
	OGBN-Arxiv	49.2 $\pm$ 2.9	53.3 $\pm$ 1.4	**56.7 $\pm$ 1.4**	52.6 $\pm$ 2.5	55.8 $\pm$ 1.4	55.8 $\pm$ 1.4

- Our **GCORN model often outperforms** existing defense approaches when subject to feature based attacks.

Results

Table: Node classification accuracy (± standard deviation) for structure-based attacks.

Attack	Dataset	GCN	GCN-Jaccard	RGCN	GNN-SVD	GNN-Guard	ParsevalR	GCORN
Mettack	Cora	73.0 ± 0.7	75.4 ± 1.8	69.2 ± 0.3	73.6 ± 0.9	74.4 ± 0.8	71.9 ± 0.7	**77.3 ± 0.5**
	CiteSeer	63.2 ± 0.9	69.5 ± 1.9	68.9 ± 0.6	65.8 ± 0.6	68.8 ± 1.5	68.3 ± 0.8	**73.7 ± 0.3**
	PubMed	60.7 ± 0.7	62.9 ± 1.8	65.1 ± 0.4	82.1 ± 0.8	**84.8 ± 0.3**	69.5 ± 1.1	71.8 ± 0.4
	CoraML	73.1 ± 0.6	75.4 ± 0.4	77.1 ± 1.1	71.3 ± 1.0	76.5 ± 0.7	76.9 ± 1.3	**79.2 ± 0.6**
PGD	Cora	76.7 ± 0.9	78.3 ± 1.1	72.0 ± 0.3	71.6 ± 0.4	75.0 ± 2.0	78.4 ± 1.2	**79.9 ± 0.4**
	CiteSeer	67.8 ± 0.8	70.9 ± 1.0	62.2 ± 1.8	60.3 ± 2.4	68.9 ± 2.2	70.6 ± 1.0	**73.1 ± 0.5**
	PubMed	75.3 ± 1.6	73.8 ± 1.3	78.6 ± 0.4	81.9 ± 0.4	**84.3 ± 0.4**	77.3 ± 0.7	77.4 ± 0.4
	CoraML	76.9 ± 1.2	75.0 ± 2.4	77.5 ± 0.3	73.1 ± 0.5	75.5 ± 0.8	81.3 ± 0.4	**84.1 ± 0.2**
DICE	Cora	74.9 ± 0.8	76.9 ± 0.9	79.6 ± 0.3	72.2 ± 1.4	75.6 ± 1.1	**79.7 ± 0.8**	78.9 ± 0.4
	CiteSeer	64.1 ± 0.5	66.0 ± 0.6	68.7 ± 0.5	62.6 ± 1.2	65.5 ± 1.1	68.9 ± 0.4	**74.6 ± 0.4**
	PubMed	79.4 ± 0.4	78.3 ± 0.2	**79.8 ± 0.4**	76.6 ± 0.5	77.8 ± 0.7	79.2 ± 0.3	78.1 ± 0.6
	CoraML	78.3 ± 0.6	77.5 ± 0.3	80.1 ± 0.4	58.7 ± 0.4	77.5 ± 0.2	80.5 ± 1.3	**81.1 ± 0.8**

- Our **GCORN model often outperforms** existing defense approaches when subject to feature based attacks.
- GCORN is also effective against **structure-based, as well as combined structure and feature attacks**.

2) A Simple and Yet Fairly Effective Defense for Graph Neural Networks

Ennadir, et al. (2024, AAAI)

Problem Set-Up

Problem: Available defense methods often have high computational complexity and training time (often increasing with increasing graph size).

Problem Set-Up

Problem: Available defense methods often have high computational complexity and training time (often increasing with increasing graph size).

Solution Approach: We propose a GNN, called the *NoisyGNN*, in which **hidden states are perturbed** by random noise following a normal distribution $N \sim \mathcal{N}(0, \beta I)$, i.e., our GNNs are of the form

$$\hat{y} = \sigma\left(\tilde{A}\,\mathrm{ReLU}\left(\tilde{A}XW^{(1)} + N\right)W^{(2)}\right).$$

Theoretical Results

Upper Bounds on GNN Vulnerability

We consider structural perturbations of the input graph (A, X), with a budget ϵ and 2-layer GNNs with 1-Lipschitz continuous activation functions and weight matrices $W^{(1)}, W^{(2)}$.

- Then, the vulnerability of GCNs is upper bounded by

$$\frac{2(\|W^{(2)}\|\|W^{(1)}\|\|X\|\epsilon)^2}{\beta};$$

- Then, the vulnerability of GINs is upper bounded by

$$\frac{(\|W^{(2)}\|\|W^{(1)}\|\|X\|\epsilon(2\|A\|+\epsilon))^2}{2\beta}.$$

Theoretical Results

Upper Bounds on GNN Vulnerability

We consider structural perturbations of the input graph (A, X), with a budget ϵ and 2-layer GNNs with 1-Lipschitz continuous activation functions and weight matrices $W^{(1)}, W^{(2)}$.

- Then, the vulnerability of GCNs is upper bounded by

$$\frac{2(\|W^{(2)}\|\|W^{(1)}\|\|X\|\epsilon)^2}{\beta};$$

- Then, the vulnerability of GINs is upper bounded by

$$\frac{(\|W^{(2)}\|\|W^{(1)}\|\|X\|\epsilon(2\|A\|+\epsilon))^2}{2\beta}.$$

Insight: Our upper bound on the vulnerability of a GNN is **smaller for large** β yielding a **more robust GNN.**

Experimental Results

Dataset	Attack Budget	GCNGuard	GCN-Jaccard	GCN-SVD	RGNN	NoisyGCN
Cora	Clean	77.5 ± 0.7	80.9 ± 0.7	80.6 ± 0.4	**83.5 ± 0.3**	83.2 ± 0.4
	Budget (5%)	75.8 ± 0.6	78.9 ± 0.8	78.4 ± 0.6	78.3 ± 0.6	**81.2 ± 0.7**
	Budget (10%)	74.7 ± 0.4	**76.7 ± 0.7**	71.5 ± 0.8	70.7 ± 0.8	74.5 ± 0.6
CiteSeer	Clean	70.1 ± 1.5	71.2 ± 0.7	70.7 ± 0.4	**72.3 ± 0.5**	71.9 ± 0.4
	Budget (5%)	69.9 ± 1.1	70.3 ± 2.3	68.9 ± 0.7	70.6 ± 0.7	**72.3 ± 0.6**
	Budget (10%)	70.0 ± 1.5	67.5 ± 2.1	68.8 ± 0.6	68.7 ± 1.2	**70.4 ± 0.8**
PubMed	Clean	84.5 ± 0.6	85.0 ± 0.5	82.7 ± 0.3	**85.1 ± 0.8**	85.0 ± 0.6
	Budget (5%)	**84.3 ± 0.9**	79.6 ± 0.3	81.3 ± 0.6	81.1 ± 0.7	81.8 ± 0.4
	Budget (10%)	**84.1 ± 0.3**	67.4 ± 1.1	81.1 ± 0.7	65.2 ± 0.4	73.3 ± 0.6
PolBlogs	Clean	93.1 ± 0.6	-	86.5 ± 0.8	94.9 ± 0.3	**95.2 ± 0.4**
	Budget (5%)	72.8 ± 0.8	-	**85.1 ± 1.6**	76.0 ± 0.8	79.7 ± 0.6
	Budget (10%)	68.7 ± 1.0	-	**84.8 ± 2.3**	69.2 ± 1.2	73.4 ± 0.5

Table: Node classification accuracy (± standard deviation) when subject to Mettack.

- Our NoisyGCNs **sometimes outperform** other defense methods.

Experimental Results

Table: Mean training time analysis (in s) of the NoisyGNN in comparison to other baselines for both the GCN and GIN instances.

Dataset	GCNGuard	GCN-Jaccard	RGCN	GCN-SVD	NoisyGCN
Cora	28.52	1.93	1.16	1.39	1.29
CiteSeer	36.04	1.58	1.23	1.12	1.24
PubMed	731.26	12.27	34.19	4.60	2.41
PolBlogs	18.17	5.17	0.96	0.80	0.65

Dataset	GINGuard	GIN-Jaccard	RGCN	GIN-SVD	NoisyGIN
Cora	48.93	3.12	1.31	1.51	1.93
CiteSeer	58.45	3.78	1.44	2.20	2.76
PubMed	963.58	16.28	41.09	6.33	7.86
PolBlogs	43.7	5.52	0.95	3.71	3.16

- Our NoisyGCNs **sometimes outperform** other defense methods.
- NoisyGNNs are **faster to train** than most other defense methods.

Experimental Results

Table: Classification accuracy (± standard deviation) of combining defense methods with the proposed noise injection on different benchmark datasets.

Method	Cora	CiteSeer	PolBlogs
GINGuard	61.8±0.5	55.6±1.8	82.7±0.6
+ Noisy	**66.2±1.3**	**58.3±1.9**	**83.6±0.8**
GIN-Jaccard	70.4±1.1	61.2±2.3	-
+ Noisy	**72.9±0.8**	**64.9±1.8**	-
GCNGuard	69.5±0.7	66.2±0.6	64.7±0.8
+ Noisy	**72.4±1.2**	**68.9±0.9**	**65.8±1.3**
GCN-Jaccard	66.7±0.5	61.2±1.1	-
+ Noisy	**69.6±0.9**	**63.1±0.6**	-

- Our NoisyGCNs **sometimes outperform** other defense methods.
- NoisyGNNs are **faster to train** than most other defense methods.
- When **combined with other defense methods**, best performance is achieved.

Conclusions

- Graph Representation Learning is a highly active area of research at the moment gaining both academic and industrial interest.

Conclusions

- Graph Representation Learning is a highly active area of research at the moment gaining both academic and industrial interest.
- Graph Neural Networks are neural networks that process graph-structured data. They are a versatile and powerful tool. And their robustness is a topic attracting much recent research.

Conclusions

- Graph Representation Learning is a highly active area of research at the moment gaining both academic and industrial interest.
- Graph Neural Networks are neural networks that process graph-structured data. They are a versatile and powerful tool. And their robustness is a topic attracting much recent research.

Specifically, with regards to the different approaches to message passing

- If the original graph is sufficient for the performed learning task, then we should simply aggregate over the fixed graph.

Conclusions

- Graph Representation Learning is a highly active area of research at the moment gaining both academic and industrial interest.
- Graph Neural Networks are neural networks that process graph-structured data. They are a versatile and powerful tool. And their robustness is a topic attracting much recent research.

Specifically, with regards to the different approaches to message passing

- If the original graph is sufficient for the performed learning task, then we should simply aggregate over the fixed graph.
- If the node features contain complementary information to the graph on the relevance of neighbours to a given node, then we should use a feature-dependent reweighting scheme on the edges.

Conclusions

- Graph Representation Learning is a highly active area of research at the moment gaining both academic and industrial interest.
- Graph Neural Networks are neural networks that process graph-structured data. They are a versatile and powerful tool. And their robustness is a topic attracting much recent research.

Specifically, with regards to the different approaches to message passing

- If the original graph is sufficient for the performed learning task, then we should simply aggregate over the fixed graph.
- If the node features contain complementary information to the graph on the relevance of neighbours to a given node, then we should use a feature-dependent reweighting scheme on the edges.
- If the graph structure is insufficient for the learning task, e.g., necessary information presents itself as a long-range effect on the original graph, then we should pick a criterion to limit the search space of n^2 node pairs and rewire accordingly.

Conclusions

- Graph Representation Learning is a highly active area of research at the moment gaining both academic and industrial interest.
- Graph Neural Networks are neural networks that process graph-structured data. They are a versatile and powerful tool. And their robustness is a topic attracting much recent research.

Specifically, with regards to the different approaches to message passing

- If the original graph is sufficient for the performed learning task, then we should simply aggregate over the fixed graph.
- If the node features contain complementary information to the graph on the relevance of neighbours to a given node, then we should use a feature-dependent reweighting scheme on the edges.
- If the graph structure is insufficient for the learning task, e.g., necessary information presents itself as a long-range effect on the original graph, then we should pick a criterion to limit the search space of n^2 node pairs and rewire accordingly.
- If we are aware of certain substructures of particular relevance to our learning task or require a highly expressive model, then we should explicitly represent these substructures in the message passing scheme.

Thank you for your attention!

johanneslutzeyer.com

References

Y. Abbahaddou, S. Ennadir, J. F. Lutzeyer, M. Vazirgiannis & H. Boström, "Bounding the Expected Robustness of Graph Neural Networks Subject to Node Feature Attacks," *International Conference on Learning Representations(ICLR),* 2024.

L. A. Adamic & N. Glance, "The political blogosphere and the 2004 US election: divided they blog," *In Proceedings of the 3rd International Workshop on Link Discovery*, pp. 36–43, 2005.

U. Alon & E. Yahav, "On the Bottleneck of Graph Neural Networks and its Practical Implications," In: *International Conference on Learning Representations (ICLR)*, 2020.

A. Bojchevski, J. Gasteiger, B. Perozzi, A. Kapoor, M. Blais, B. Rózemberczki, M. Lukasik & S. Günnemann, "Scaling graph neural networks with approximate pagerank," *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 2464-2473, 2020.

F. Borisyuk, S. He, Y. Ouyang, M. Ramezani, P. Du, X. Hou, C. Jiang, N. Pasumarthy, P. Bannur, B. Tiwana, P. Liu, "LiGNN: Graph Neural Networks at LinkedIn," *arXiv:2402.11139*, 2024.

S. Brody, U. Alon & E. Yahav, "How Attentive Are Graph Attention Networks?," In: *International Conference on Learning Representations (ICLR)*, 2022.

M. Bronstein, "Graph ML at Twitter," *Twitter Engineering Blog Post*, `https://blog.twitter.com/engineering/en_us/topics/insights/2020/graph-ml-at-twitter`, 2020.

M. Bronstein, "Geometric Deep Learning: The Erlangen Programme of ML," *Keynote Talk at The International Conference on Learning Representations*, 2021.

S. Butler & F. Chung, "Spectral graph theory," In: L. Hogben (ed) *Handbook of linear algebra (2nd edition)*, Boca Raton, FL: CRC Press, pp. 47/1—47/14, 2017.

M. Chatzianastasis, J. F. Lutzeyer, G. Dasoulas & M. Vazirgiannis, "Graph Ordering Attention Networks," *Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI),* 2023.

D. Cvetkovic, R. Rowlinson & S. Simic, *Eigenspaces of graphs*, Cambridge, UK: Cambridge University Press, 1997.

L. Dall'Amico, R. Couillet & N. Tremblay, "Optimal Laplacian regularization for sparse spectral community detection," *ICASSP*, 2020.

G. Dasoulas, J. F. Lutzeyer & M. Vazirgiannis, "Learning Parametrised Graph Shift Operators," In: *International Conference on Learning Representations (ICLR)*, 2021.

A. Deac, M. Lackenby & P. Veličković, "Expander Graph Propagation," *arXiv:2210.02997*, 2022.

J. A. Deri & J. M. F. Moura, "Spectral projector-based graph Fourier transforms," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, pp. 785–795, 2017.

V.P. Dwivedi & X. Bresson, "A Generalization of Transformer Networks to Graphs," *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

S. Ennadir, Y. Abbahaddou, J. F. Lutzeyer, M. Vazirgiannis & H. Boström, "A Simple and Yet Fairly Effective Defense for Graph Neural Networks," *Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI),* 2024.

F. Frasca, B. Bevilacqua, M. Bronstein & H. Maron, "Understanding and Extending Subgraph GNNs by Rethinking Their Symmetries," *Proceedings of the 36st International Conference on Neural Information Processing Systems (NeurIPS),* 2022.

F. Geerts & J. L. Reutter, "Expressiveness and Approximation Properties of Graph Neural Networks," *International Conference on Learning Representations (ICLR)*, 2022.

I. J. Goodfellow, J. Shlens, & C. Szegedy, "Explaining and harnessing adversarial examples," International Conference of Learning Representations (ICLR), 2015.

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals & G. E. Dahl, "Neural message passing for Quantum chemistry," *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

S. Günnemann, "Graph Neural Networks: Adversarial Robustness," *Graph Neural Networks: Foundations, Frontiers, and Applications*, pp. 149–176, 2022.

B. Hammer, "On the approximation capability of recurrent neural networks," *Neurocomputing*, pp. 107–123, 2000.

A. Jain, I. Liu, A. Sarda & P. Molino, "Food Discovery with Uber Eats: Using Graph Learning to Power Recommendations," *Uber Engineering Blog Post*, `https://eng.uber.com/uber-eats-graph-learning/`, 2019.

Thomas N. Kipf & M. Welling, "Variational Graph Auto-Encoders" *NeurIPS Workshop on Bayesian Deep Learning*, 2016.

Thomas N. Kipf & M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations (ICLR)*, 2017.

O. Lange & L. Perez, "Traffic prediction with advanced Graph Neural Networks," *DeepMind Research Blog Post*, `https://deepmind.com/blog/article/traffic-prediction-with-advanced-graph-neural-networks`, 2020.

G. Liu, D. B. Catacutan, K. Rathod, K. Swanson, W. Jin, J. C. Mohammed, A. Chiappino-Pepe, S. A. Syed, M. Fragis, K. Rachwalski, J. Magolan, M. G. Surette, B. K. Coombes, T. Jaakkola, R. Barzilay, J. J. Collins, J. M. Stokes, "Deep learning-guided discovery of an antibiotic targeting Acinetobacter baumannii," *Nature Chemical Biology*, pp. 1–9, 2023.

J. Lutzeyer, *Network Representation Matrices and their Eigenproperties: A Comparative Study,* PhD thesis: Imperial College London, 2020.

J. Lutzeyer, C. Wu & M. Vazirgiannis, "Graph Neural Network Simplification: Sparsifying the Update Step," *ICLR Workshop on Geometrical and Topological Representation Learning*, 2022.

G. Michel, G. Nikolentzos, J. Lutzeyer & M. Vazirgiannis, "Path Neural Networks: Expressive and Accurate Graph Neural Networks," *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J.E Lenssen, G. Rattan & M. Grohe, "Weisfeiler and Lehman Go Neural: Higher-order Graph Neural Networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 4602–4609, 2019.

A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura & P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, pp. 808–828, 2018.

T. Qin & K. Rohe, "Regularized Spectral Clustering under the Degree-Corrected Stochastic Blockmodel," *Advances in neural information processing systems (NIPS)*, pp. 3120–3128, 2013.

A. R. Ramos Vela, J. F. Lutzeyer, A. Giovanidis & M. Vazirgiannis, "Improving Graph Neural Networks at Scale: Combining Approximate PageRank and CoreRank," *NeurIPS New Frontiers in Graph Learning Workshop*, 2022.

G. Salha-Galvan, J. F. Lutzeyer, G. Dasoulas, R. Hennequin & M. Vazirgiannis, "Modularity-Aware Graph Autoencoders for Joint Community Detection and Link Prediction," *arxiv:2202.00961*, 2022.

G. Salha-Galvan, *Contributions to Representation Learning with Graph Autoencoders and Applications to Music Recommendation,* PhD thesis: École Polytechnique, Institut Polytechnique de Paris, 2022.

A. Sandryhaila & J. M. F. Moura "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, pp. 1644–1656, 2013.

M. E. A. Seddik, C. Wu, J. F. Lutzeyer & M. Vazirgiannis, "Node Feature Kernels Increase Graph Convolutional Network Robustness," *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher & T. Eliassi-Rad, *AI Magazine*, p. 93, 2008.

J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, V. M. Tran, A. Chiappino-Pepe, A. H. Badran, I. W. Andrews, E. J. Chory, G. M. Church, E. D. Brown, T. S. Jaakkola, R. Barzilay & J. J. Collins, "A Deep Learning Approach to Antibiotic Discovery," *Cell*, pp. 688–702, 2020.

L. Sun, Y. Dou, C. Yang, J. Wang, P. S. Yu & B. Li, "Adversarial attack and defense on graph data: A survey," *arXiv:1812.10528*, 2020.

J. Topping, F. Di Giovanni, B. P. Chamberlain, X. Dong & M. M. Bronstein, "Understanding Over-Squashing and Bottlenecks on Graphs via Curvature," *International Conference on Learning Representations (ICLR)*, 2022.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò & Y. Bengio, "Graph Attention Networks," *6th International Conference on Learning Representations (ICLR),* 2018.

P. Veličković, "Message Passing All the Way Up," *ICLR Workshop on Geometrical and Topological Representation Learning,* 2022.

S. Virinchi, A. Saladi & A. Mondal, "Recommending Related Products Using Graph Neural Networks in Directed Graphs," In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2022.

P. L. Williams & R. D. Beer, "Nonnegative decomposition of multivariate information," *arXiv:1004.2515*, 2010.

J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao & Dik Lun Lee, "Billion-scale Commodity Embedding for E-Commerce Recommendation in Alibaba," In Proceedings of the *24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pp. 839–848, 2018.

K. Xu, W. Hu, J. Leskovec & S. Jegelka."How powerful are graph neural networks?", *International Conference on Learning Representations (ICLR)*, 2019.

R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton & J. Leskovec, "Graph Convolutional Neural Networks for Web-Scale Recommender Systems," In Proceedings of the *24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pp. 974–983, 2018.

Y. Zhou, H. Zheng & X. Huang, "Graph Neural Networks: Taxonomy, Advances and Trends," *arXiv:2012.08752*, 2020.