Graph Shift Operators and Their Relevance to Graph Neural Networks

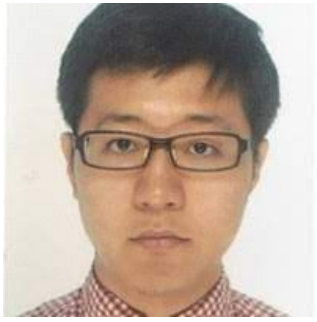
Johannes Lutzeyer

Data Science and Mining Team, Laboratoire d'Informatique,
École Polytechnique, Institut Polytechnique de Paris

March 8, 2022

Today I present work was done in collaboration with:

George Dasoulas
CIFRE PhD LIX & Huawei



Dr. Changmin Wu
Postdoctoral Researcher LIX

Dr. Guillaume Salha Galvan
CIFRE PhD LIX & Deezer

Dr. Mohamed E. A. Seddik
Researcher Huawei

Dr. Romain Hennequin
Lead Research Scientist
Deezer

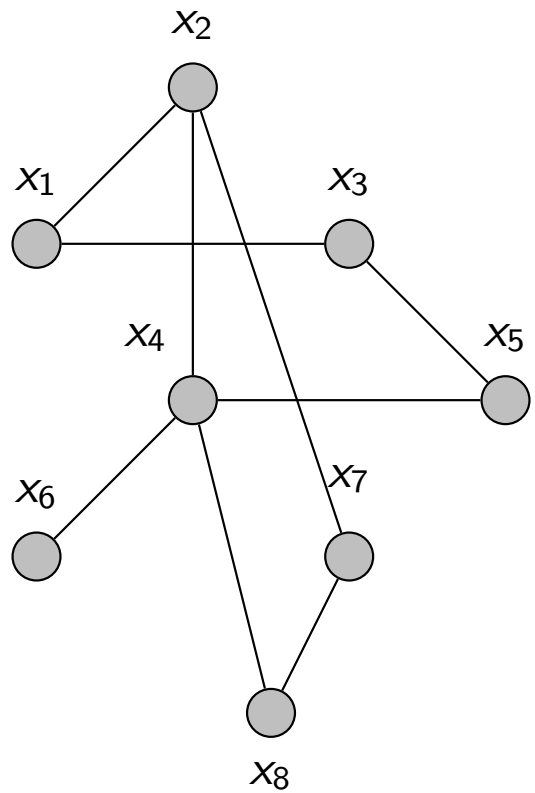
Prof. Michalis Vazirgiannis
Distinguished Professor LIX

Graph Neural Networks

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

We consider graph-structured data to be the combination of

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



In this talk we will only see a specific type of GNN, the Message Passing Neural Networks.

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

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

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

Other examples of popular GNN architectures are the GIN (Xu et al., 2019), GraphSage (Hamilton et al., 2017) and GAT (Veličković et al., 2018).

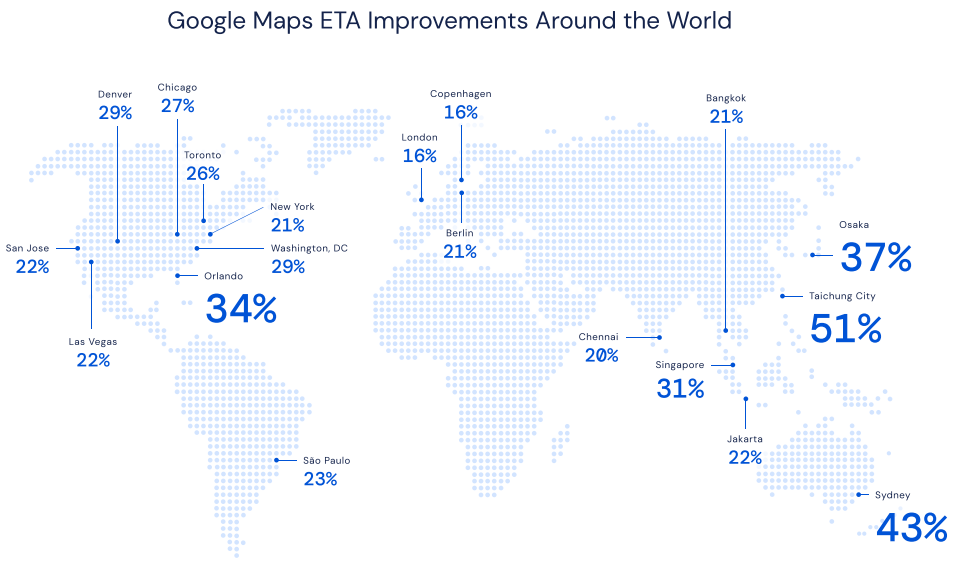
Academic and Industrial Success of GNNs

Rich ground for empirical and theoretical research

- expressivity analysis of different message passing operators (Xu et al., 2019; Morris et al., 2019);
- analysis of their robustness to adversarial attacks and noise (Günnemann, 2022; Sun et al., 2020; Zhou et al., 2020).

Successful applications of GNNs:

- *Google*: Improved Estimated Time of Arrival estimation in Google Maps (Lange and Perez, 2020);
- *Twitter*: Fake news detection (Bronstein, 2020);
- even helped discover a *new antibiotic* (Stokes et al., 2020).

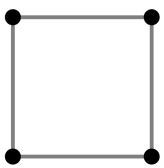


Graph Shift Operators

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* $L_{rw} = D^{-1} L$.



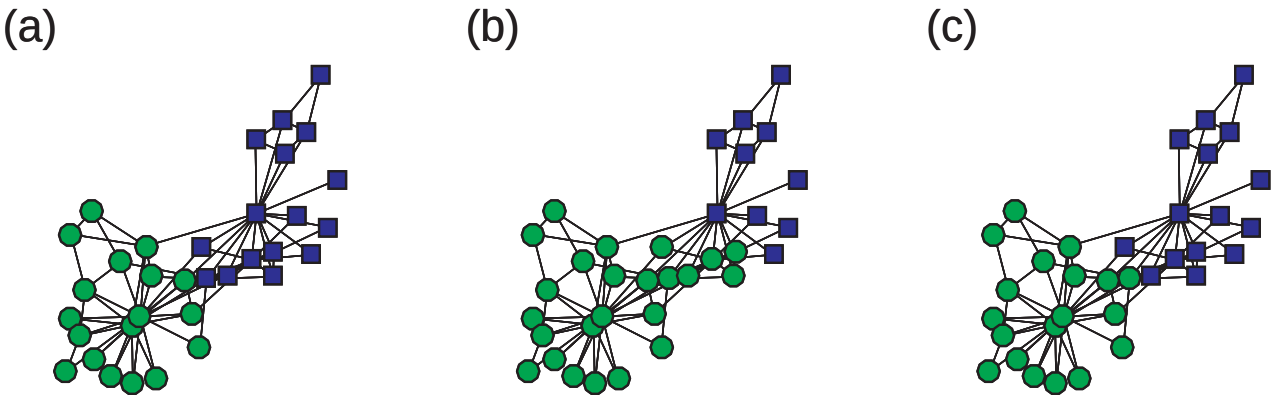
$$A = \begin{pmatrix} 0&1&0&1 \\ 1&0&1&0 \\ 0&1&0&1 \\ 1&0&1&0 \end{pmatrix} \quad L = \begin{pmatrix} 2&-1&0&-1 \\ -1&2&-1&0 \\ 0&-1&2&-1 \\ -1&0&-1&2 \end{pmatrix} \quad L_{sym} = \begin{pmatrix} 1&-0.5&0&-0.5 \\ -0.5&1&-0.5&0 \\ 0&-0.5&1&-0.5 \\ -0.5&0&-0.5&1 \end{pmatrix}$$

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 $(i,j) \notin E$.

GSOs in 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.

Overview of the Talk

Recall,

- GNN: neural networks that process graph-structured data
- GSO: matrices that represent graphs

In this talk,

1) learn optimal GSO in GNNs;
2) introduce node feature information to GSO to robustify GNNs;
3) introduce global cluster information to GSO in Graph Autoencoders.

1) Learning Parametrised Graph Shift Operators

Dasoulas, Lutzeyer & Vazirgiannis (2021, ICLR)

Motivation to Learn GSOs

- When introducing the different standard GSO choices Butler and Chung (2017) state: *"No one matrix is best because each matrix has its own limitations in that there is some property which the matrix cannot always determine"*.
- Graph signal processing literature: the GSO choice involves "different tradeoffs" and leads to different signal models (Deri and Moura, 2017; Ortega et al., 2018). Therefore, they recommend using *whichever GSO works best* in a particular analysis or learning task.

Research Questions

Question 1: *Is there a single optimal representation to encode graph structures or is the optimal representation task- and data-dependent?*
Question 2: *Can we learn such an optimal representation to encode graph structure in a numerically stable and computationally efficient way?*

Parametrised Graph Shift Operators

Definition

We define the *Parametrised Graph Shift Operator (PGSO)*, denoted by $\gamma(A, \mathcal{S})$, as

$$\gamma(A, \mathcal{S}) = m_1 D_a^{e_1} + m_2 D_a^{e_2} A_a D_a^{e_3} + m_3 I_n, \tag{2}$$

where $A_a = A + aI_n$, $D_a = \text{Diag}(A_a \mathbf{1}_n)$ and $S = (m_1, m_2, m_3, e_1, e_2, e_3, a)$.

$\mathcal{S} = (m_1, m_2, m_3, e_1, e_2, e_3, a)$	**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.
- $X \in \mathbb{R}^{n \times d}$ denotes the node feature matrix.
- $\mathcal{M}(\phi(A), X)$ denotes a GNN model.
- K number of aggregation layers in a GNN.

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

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 $\epsilon_{ij} = m_2 \left(D_a\right)_i^{e_2} \left(D_a\right)_j^{e_3}$.

Spectral Analysis

Theorem

$\gamma(A, \mathcal{S})$ has real eigenvalues and a set of real eigenvectors independent of the parameters chosen in $\mathcal{S}$.

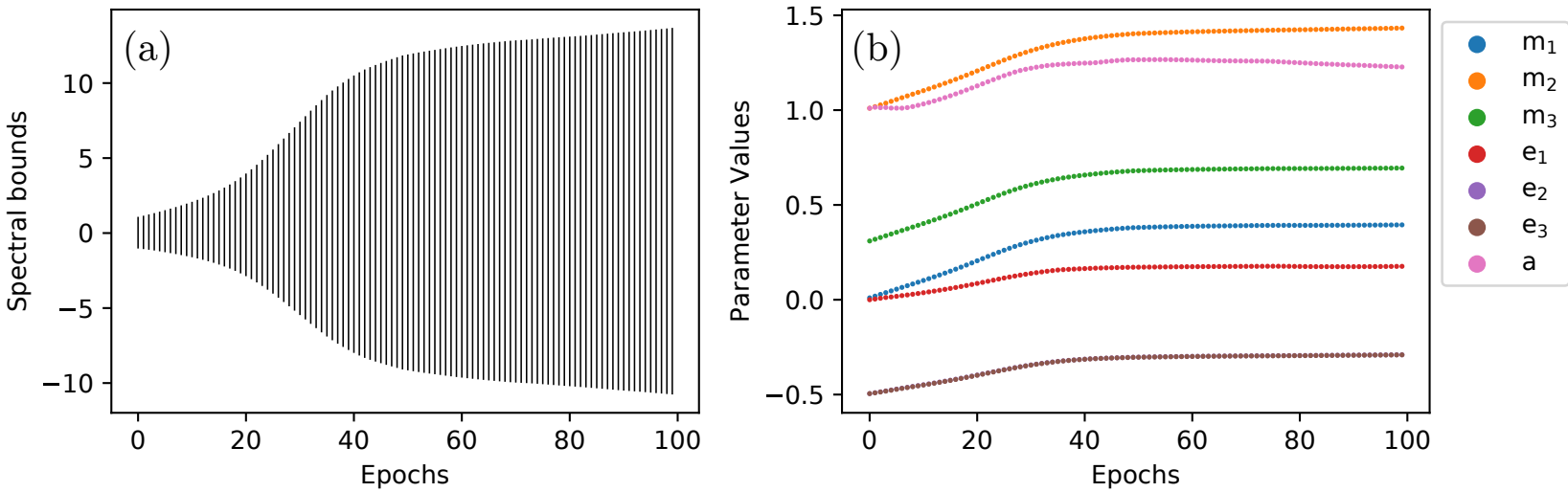
Theorem

Let $C_i = m_1(d_i + a)^{e_1} + m_2(d_i + a)^{e_2+e_3} a + m_3$ and $R_i = |m_2|(d_i + a)^{e_2+e_3} d_i$, where d_i denotes the degree of node v_i. Furthermore, we denote eigenvalues of $\gamma(A, \mathcal{S})$ by $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. Then, for all $j \in \{1, \ldots, n\}$,

$$\lambda_j \in \left[\min_{i \in \{1,\ldots,n\}} (C_i - R_i), \max_{i \in \{1,\ldots,n\}} (C_i + R_i) \right]. \tag{3}$$

- For the parametrisation of $\gamma(A, \mathcal{S})$ corresponding to the adjacency matrix, we obtain the spectral support $[-d_{\max}, d_{\max}]$, as required.
- For the message passing operator in the GCN, we obtain the following bounds on the spectral support $[-(d_{\max} - 1)/(d_{\max} + 1), 1]$, the lower bound of this interval tends to -1 as $d_{\max} \to \infty$.

Spectral Analysis: Empirical Observation

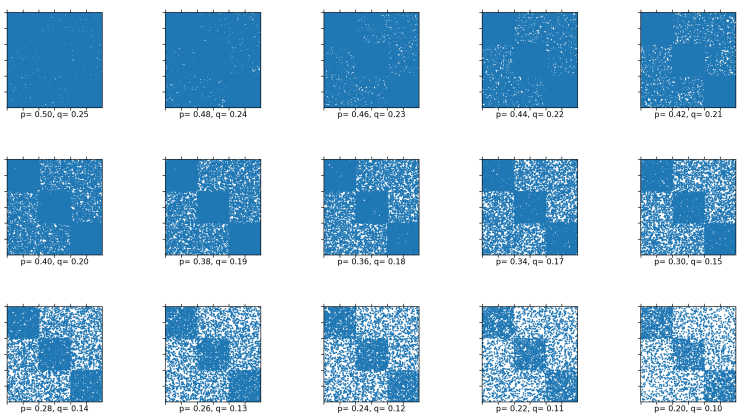


- Surprisingly, the spectral support of the PGSO remains centered at 0 throughout training.
- We observe the parameters of the PGSO to be smoothly varying throughout training.

$\gamma(A, \mathcal{S})$ on Stochastic Block Models (SBMs) of varying Sparsity

Setup:

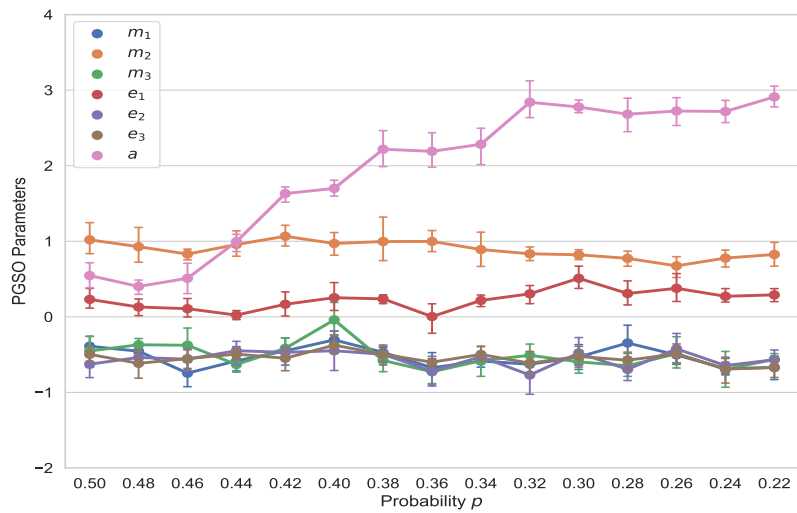
- 15 decreasing p, q combinations.
- Fixed detectability level.
- $\forall(p, q)$ 25 sampled graphs with 3 200-node communities.
- Node classification (Dwivedi et al., 2020).
- 3-layer GCN-PGSO model.



SBM adjacency matrices.

Remarks:

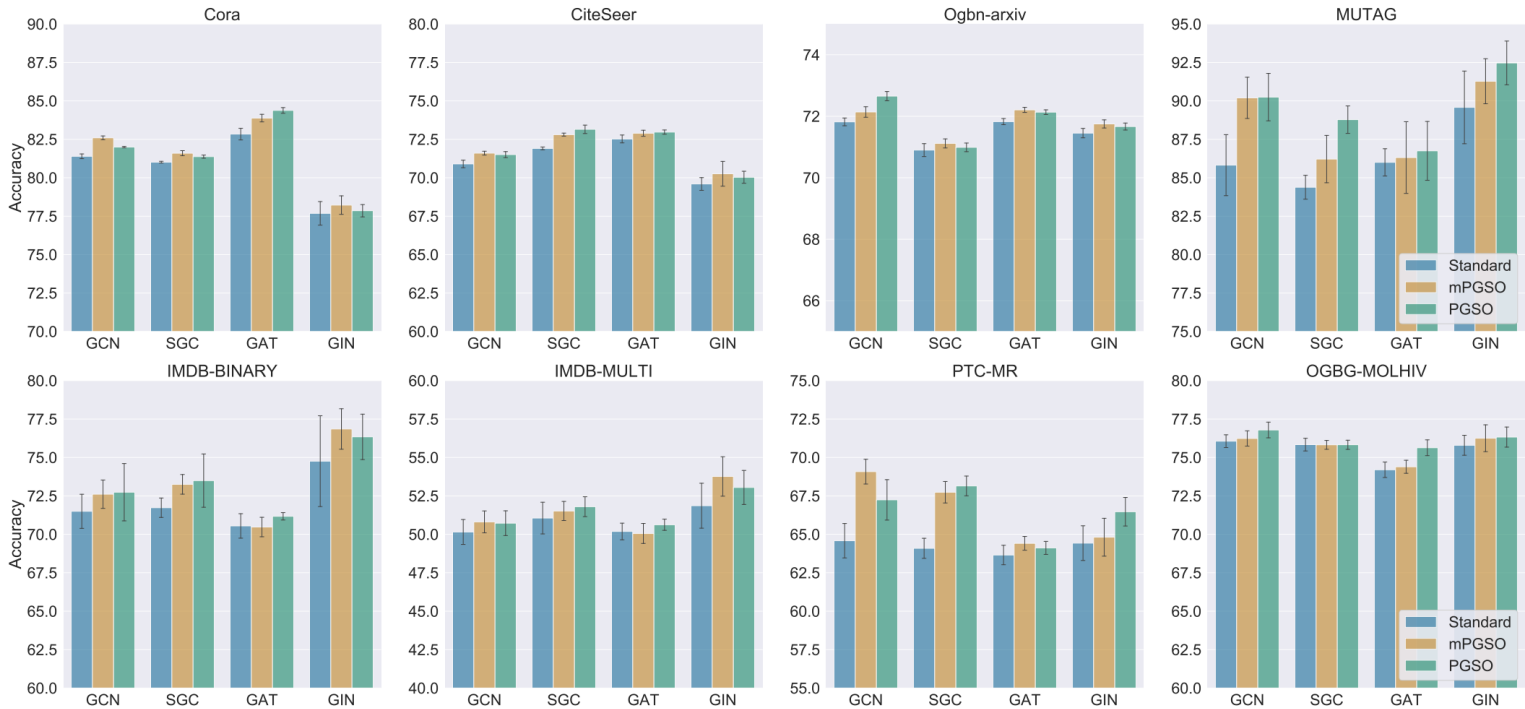
- The additive parameter a increases with the increasing sparsity.
- The remaining parameters remain close to constant.
- confirms the positive impact of GSO regularisation (Dall'Amico et al., 2020; Qin and Rohe, 2013)



Mean/std of the PGSO parameters

Results on Real-world Datasets

- Evaluation in 8 **node classification** and **graph classification** tasks.
- Model design with 4 architectures: GCN, SGC, GAT and GIN models.
- 3 GSO variants for each model: **Standard**, **PGSO** and **mPGSO**.



- 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.
- There is **no clear** winner between PGSO and mPGSO.
- Our code is publicly available: https://github.com/gdasoulas/PGSO.

2) Node Feature Kernels Increase Graph Convolutional Network Robustness

Seddik, Wu, Lutzeyer & Vazirgiannis (2022, AISTATS)

Let's begin with the Highlights

Recall, a GNN takes two inputs: the graph structure and node features.

Our Observation

If the graph is sufficiently perturbed then the GCN fails to benefit from the node features no matter how informative they are.

Intuitive explanation: Node features are aggregated over graph neighbourhoods. If these neighbourhoods are random, then we are aggregating random subsets of node features.

This can be addressed by replacing the GCN's GSO $\tilde{A} = D_1^{-\frac{1}{2}} A_1 D_1^{-\frac{1}{2}}$ by

$$\tilde{A} + \mathrm{diag}((K \circ A)\mathbf{1})^{-\frac{1}{2}}(K \circ A)\mathrm{diag}((K \circ A)\mathbf{1})^{-\frac{1}{2}},$$

where $K = XX^T$ and $\circ$ denotes the elementwise multiplication.

Further research should be done on:

1) better choices of kernel function;
2) more informative sparsification schemes.

The Random GCN

This result was obtained in a random matrix theory analysis of our one-layer *Random*GCN

$$\sigma(\tilde{A}XW) \quad \text{with} \quad W_{ij} \sim \mathcal{N}(0,1).$$

Under the assumptions that,

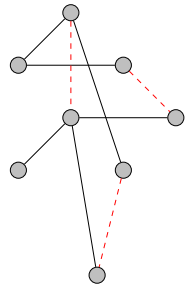
- *Node features* follow a Gaussian Mixture Model.
- *Graph Structure* follows a Stochastic Block Model (SBM).
- *Growth Rate Assumptions* on the number of nodes, feature dimension and dimension of the random matrix W and edge probabilities.
- *Regularity Assumptions* on the activation function $\sigma(\cdot)$.

we established,

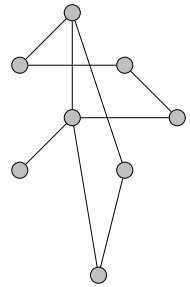
(Informal) Theorem

The extent to which the labels vector correlates with the informative eigenvector of the Gram matrix of our *Random*GCN depends on the presence of cluster structure in the SBM.

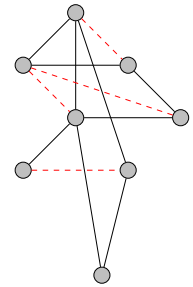
Experiments: Stochastic Blockmodels



Edge Deletion
α proportion of original edges removed



Original



Edge Insertion
β proportion of added edges

- 2-community SBMs with no community structure, weakly homophilic communities and weakly heterophilic communities.
- GCN-k denotes the GCN including the node feature kernel.

		SBM($p = 0.25$, $q = 0.25$)		SBM($p = 0.275$, $q = 0.25$)		SBM($p = 0.225$, $q = 0.25$)	
	(α, β)	GCN	GCN-k	GCN	GCN-k	GCN	GCN-k
	(0.0, 0.0)	50.53 ± 0.49	**66.36 ± 0.81**	**64.42 ± 0.43**	62.26 ± 1.04	**63.20 ± 0.94**	61.03 ± 1.08
Deletion	(0.2, 0.0)	51.03 ± 0.56	**65.44 ± 1.07**	58.63 ± 0.68	**71.57 ± 1.42**	**60.89 ± 0.83**	54.91 ± 1.00
	(0.5, 0.0)	49.29 ± 0.59	**64.14 ± 1.01**	60.76 ± 1.29	**68.80 ± 2.04**	58.41 ± 1.11	59.51 ± 2.47
Insertion	(0.0, 0.5)	50.57 ± 0.75	**68.57 ± 1.25**	60.49 ± 0.40	**68.20 ± 1.38**	58.82 ± 1.16	**63.54 ± 0.97**
	(0.0, 1.0)	49.19 ± 0.47	**59.31 ± 0.58**	53.67 ± 1.11	**66.57 ± 1.73**	54.87 ± 0.53	**60.84 ± 0.75**
Delet.+Insert.	(0.5, 0.5)	49.26 ± 0.59	**68.84 ± 0.86**	50.50 ± 0.37	**63.36 ± 1.67**	50.94 ± 0.86	**63.02 ± 0.91**
	(0.5, 1.0)	49.84 ± 0.69	**65.49 ± 1.22**	48.34 ± 0.22	**60.16 ± 1.21**	49.23 ± 0.45	**59.64 ± 1.33**

The addition of the **node feature kernel improves the GCN's robustness** against edge-deletion and edge insertion noise.

Experiments: Real World Datasets

	(α, β)	Cora GCN	Cora GCN-k	CiteSeer GCN	CiteSeer GCN-k	PubMed GCN	PubMed GCN-k
Deletion	(0.0, 0.0)	**79.37 ± 0.65**	76.94 ± 0.35	67.45 ± 0.82	68.08 ± 0.91	76.04 ± 0.67	74.68 ± 0.76
	(0.2, 0.0)	76.15 ± 0.81	74.83 ± 1.24	66.79 ± 0.57	66.94 ± 0.82	**75.82 ± 0.99**	74.28 ± 0.39
	(0.5, 0.0)	72.49 ± 0.50	71.18 ± 1.00	63.53 ± 0.75	64.84 ± 1.14	73.95 ± 0.64	73.25 ± 0.75
Insertion	(0.0, 0.5)	68.57 ± 0.73	**73.10 ± 1.10**	59.85 ± 0.89	**66.11 ± 1.34**	64.18 ± 0.67	**72.38 ± 0.79**
	(0.0, 1.0)	64.14 ± 1.02	**73.36 ± 0.98**	55.39 ± 0.93	**64.94 ± 0.77**	60.56 ± 0.80	**71.31 ± 0.51**
Delet.+Insert.	(0.5, 0.5)	54.98 ± 1.13	**66.46 ± 1.03**	52.84 ± 0.68	**59.03 ± 1.04**	62.62 ± 0.72	**70.32 ± 0.82**
	(0.5, 1.0)	48.09 ± 0.88	**62.52 ± 0.59**	42.28 ± 1.07	**58.07 ± 1.34**	53.25 ± 1.57	**69.65 ± 0.60**

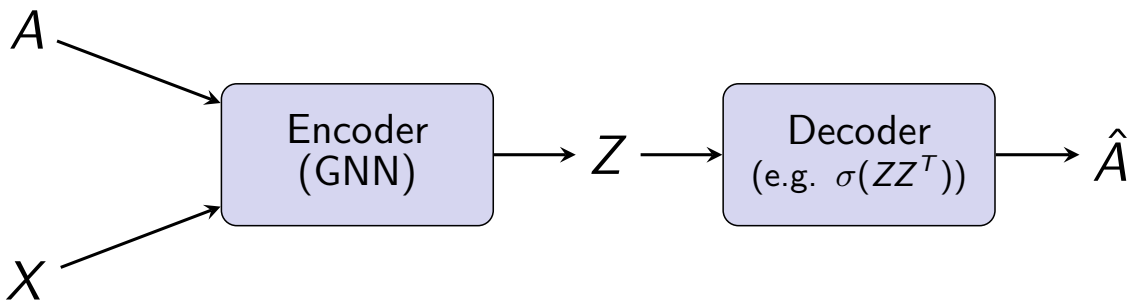
	(α, β)	CoraFull GCN	CoraFull GCN-k	Photo GCN	Photo GCN-k	CS GCN	CS GCN-k
Deletion	(0.0, 0.0)	57.21 ± 0.84	56.88 ± 0.48	90.94 ± 0.49	90.09 ± 0.65	92.89 ± 0.41	92.63 ± 0.31
	(0.2, 0.0)	**57.25 ± 0.67**	55.56 ± 0.69	91.87 ± 0.40	92.19 ± 0.45	90.58 ± 0.48	90.89 ± 0.48
	(0.5, 0.0)	53.90 ± 0.70	54.62 ± 0.87	**91.10 ± 0.40**	87.97 ± 0.54	89.75 ± 0.60	**91.27 ± 0.67**
Insertion	(0.0, 0.5)	48.11 ± 0.89	**51.79 ± 0.65**	82.79 ± 1.43	84.18 ± 1.27	87.16 ± 0.65	**90.81 ± 0.70**
	(0.0, 1.0)	41.76 ± 1.03	**51.91 ± 1.00**	72.70 ± 6.40	**79.58 ± 1.80**	80.34 ± 0.80	**90.61 ± 0.37**
Delet.+Insert.	(0.5, 0.5)	34.70 ± 0.47	**46.50 ± 0.61**	69.70 ± 3.70	**74.65 ± 2.36**	73.75 ± 0.98	**87.28 ± 0.72**
	(0.5, 1.0)	27.50 ± 1.04	**43.04 ± 0.77**	61.13 ± 2.49	63.73 ± 5.04	66.26 ± 0.95	**87.51 ± 0.58**

- On real world datasets insertion noise seems to have a greater impact, which can largely be **compensated by the node feature kernel**.
- We also observed similar behaviour for other GNN architectures (GIN, GraphSage and GAT).
- Our code is publicly available: https://github.com/ChangminWu/RobustGCN.

3) Modularity-Aware Graph Autoencoders for Joint Community Detection and Link Prediction

Salha-Galvan, Lutzeyer, Dasoulas, Hennequin & Vazirgiannis (2022)

Graph Autoencoders (GAEs)



Graph Autoencoders (GAEs):

- typically **consist of** the composition of a GNN with an inner product decoder reconstructing the graph structure.
- **learn** low-dimensional representations Z in an unsupervised manner.
- currently find **industrial use in recommendation systems**.

Motivation of our Project

Research Problem

Graph Autoencoders are very good at link prediction and often underwhelming in community detection. Learning node embeddings Z that enable good performance in both tasks is desirable for real-world applications.

Our Modularity-Aware Graph Autoencoders address this problem by

- using a different GSO in the encoder's message passing scheme.
- modifying the loss function to consider a softened version of the modularity.
- considering both the clustering's modularity and the classification AUC in the hyperparameter selection.

To stay on topic, we will discuss only the first of these contributions.

Introducing Global Cluster Information to the Message Passing Step

Steps of our method modifying the encoder (GNN):

1) We run the Louvain algorithm to cluster the graph.
   - Automatically determines the number of communities.
   - It's fast $O(n \log n)$.
   - It maximises the modularity, which complements our other contributions.
2) We define a graph with adjacency A_c composed of fully connected components corresponding to the communities obtained in step 1).
3) We randomly sample s neighbours for each node in its fully connected component to define a subgraph represented by A_s.
4) We replace A in the GNN encoder by

$$A + \lambda A_s,$$

where $\lambda \geq 0$ is a scalar hyperparameter determining the importance of the cluster information.

Experiments: Cora Graph without Node Features

A weakness of existing models addressing this problem:
Their performance heavily decreases if no node features are available.

Models (Dimension $d = 16$)	**Joint Link Prediction and Community Detection on graph with 15% of edges being masked**			
	AMI (in %)	**ARI (in %)**	**AUC (in %)**	**AP (in %)**
Modularity-Aware GAE/VGAE Models				
Linear Modularity-Aware VGAE	*42.86 ± 1.65*	*34.53 ± 1.97*	*85.96 ± 1.24*	*87.21 ± 1.39*
Linear Modularity-Aware GAE	**43.48 ± 1.12**	**35.51 ± 1.20**	**87.18 ± 1.05**	*88.53 ± 1.33*
GCN-based Modularity-Aware VGAE	41.03 ± 1.55	*33.43 ± 2.17*	84.87 ± 1.14	85.16 ± 1.23
GCN-based Modularity-Aware GAE	41.13 ± 1.35	*35.01 ± 1.58*	*86.90 ± 1.16*	*87.55 ± 1.26*
Standard GAE/VGAE Models				
Linear VGAE	32.22 ± 1.76	21.82 ± 1.80	85.69 ± 1.17	**89.12 ± 0.82**
Linear GAE	28.41 ± 1.68	19.45 ± 1.75	84.46 ± 1.64	*88.42 ± 1.07*
GCN-based VGAE	28.62 ± 2.76	19.70 ± 3.71	85.47 ± 1.18	*88.90 ± 1.11*
GCN-based GAE	31.30 ± 2.07	19.89 ± 3.07	85.31 ± 1.35	*88.67 ± 1.24*
Other Baselines				
Louvain	39.09 ± 0.73	20.19 ± 1.73	–	–
VGAECD	33.54 ± 1.46	24.32 ± 2.25	83.12 ± 1.11	84.68 ± 0.98
VGAECD-OPT	34.41 ± 1.62	24.66 ± 1.98	82.89 ± 1.20	83.70 ± 1.16
ARGVA	28.96 ± 2.64	19.74 ± 3.02	85.85 ± 0.87	*88.94 ± 0.72*
ARGA	31.61 ± 2.05	20.18 ± 2.92	85.95 ± 0.85	*89.07 ± 0.70*
DVGAE	30.46 ± 4.12	21.06 ± 5.06	85.58 ± 1.31	*88.77 ± 1.29*
DeepWalk	30.26 ± 2.32	20.24 ± 3.91	80.67 ± 1.50	80.48 ± 1.28
node2vec	36.25 ± 1.38	29.43 ± 2.21	82.43 ± 1.23	81.60 ± 0.91

In the absence of node features, our model outperforms a large number of baselines achieving good performance in both tasks.

Experiments: Real-World Datasets with Node Features

Datasets	**Models** (Dimension $d = 16$)	**Joint Link Prediction and Community Detection on graph with 15% of edges being masked**			
		AMI (in %)	**ARI (in %)**	**AUC (in %)**	**AP (in %)**
Cora	Linear Modularity-Aware VGAE	**49.70 ± 2.04**	**43.64 ± 3.51**	**93.10 ± 0.88**	**94.06 ± 0.75**
	Linear Standard VGAE	46.90 ± 1.43	38.24 ± 3.56	*93.04 ± 0.80*	*94.04 ± 0.75*
	Louvain	39.09 ± 0.73	20.19 ± 1.73	–	–
	Best other baseline:				
	VGAECD-OPT	47.83 ± 1.64	39.45 ± 3.53	*92.25 ± 1.07*	92.60 ± 0.91
Citeseer	Linear Modularity-Aware VGAE	*22.21 ± 1.24*	**12.59 ± 1.25**	86.54 ± 1.20	88.07 ± 1.22
	Linear Standard VGAE	17.38 ± 1.43	6.10 ± 1.51	**89.08 ± 1.19**	**91.19 ± 0.98**
	Louvain	**22.71 ± 0.47**	7.70 ± 0.67	–	–
	Best other baseline:				
	DVGAE	16.02 ± 3.32	*10.03 ± 4.48*	86.85 ± 1.48	88.43 ± 1.23
Deezer-Album	GCN-Based Modularity-Aware VGAE	**19.10 ± 0.21**	**12.00 ± 0.17**	**85.40 ± 0.14**	*86.38 ± 0.15*
	GCN-Based Standard VGAE	13.98 ± 0.35	8.81 ± 0.32	*85.37 ± 0.12*	**86.41 ± 0.11**
	Louvain	17.68 ± 0.20	11.02 ± 0.13	–	–
	Best other baseline:				
	node2vec	18.34 ± 0.29	11.27 ± 0.28	83.51 ± 0.17	84.12 ± 0.15

The good performance of our model extends to industrial scale datasets such as a private Deezer graph containing 2.5 million music albums and 25 million edges.

Conclusions

- Learning optimal graph representation – via a Parametrised GSO – in GNNs improves their performance on real world datasets.
- Node feature information only informs GNN inference if an informative graph structure is present. Adding node feature kernels to the GSO allows GNNs to consider their two input sources in a more balanced manner.
- Adding global cluster information to the GSO in GNNs improves the performance of Graph Autoencoders in industrial application.

We are currently looking for Postdocs!

Please get in touch if you are interested in working on
Natural Language Processing or *Graph Representation Learning*!

Thank you for your attention!

 @JLutzeyer

References

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.

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.

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.

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, C.K. Joshi, T. Laurent, Y. Bengio, X. Bresson, "Benchmarking Graph Neural Networks," *arXiv:2003.00982*, 2020.

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.

W. L. Hamilton, R. Ying & J. Leskovec, "Inductive Representatino Learning on Large Graphs," *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS),* pp. 1025 – 1035, 2017.

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.

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

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.

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.

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.

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.

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

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

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