Graph Neural Networks

Operators and Architectures

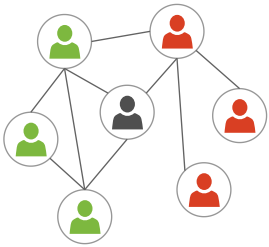
Daniele Grattarola

Dissertation defence - December 10, 2021

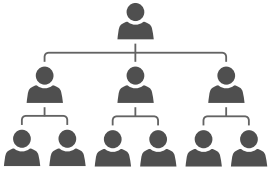
Advisor: Prof. Alippi
Co-advisor: Prof. Livi (U. Manitoba)

Internal committee: Profs. Crestani, Gambardella
External committee: Profs. Angelov (Lancaster U.), Panayiotou (U. Cyprus), Sperduti (U. Padova)

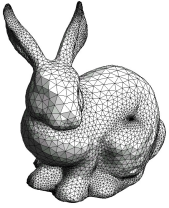
Graphs are everywhere



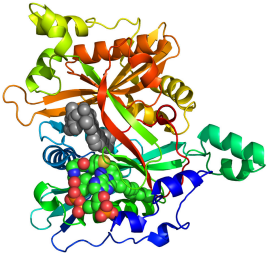
Social networks



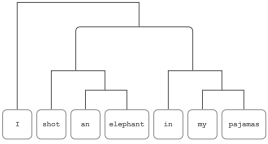
Engineering



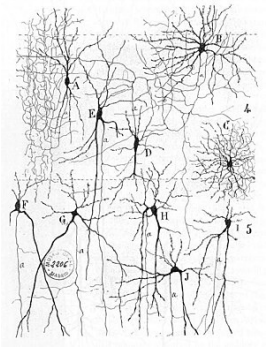
Computer vision



Biology

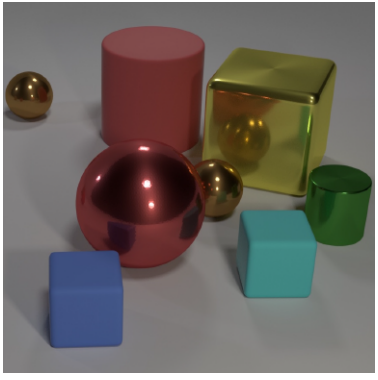


Language



Neuroscience

Graphs are everywhere

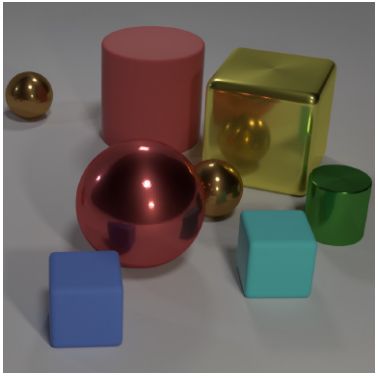


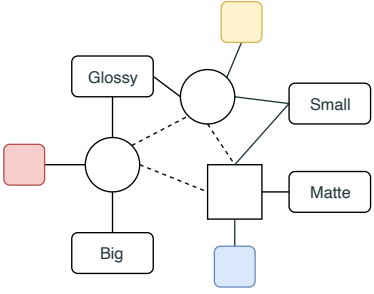
Combinatorial generalisation

Combining known concepts to represent new ones.

[1] P. W. Battaglia *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

Graphs are everywhere



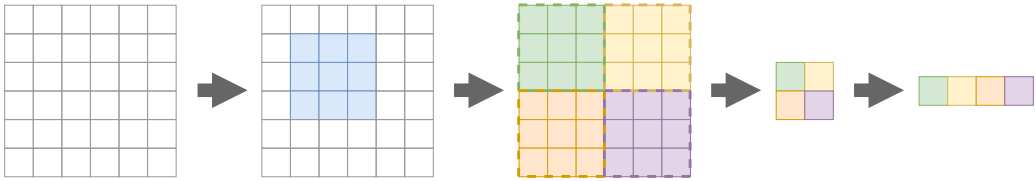


Combinatorial generalisation

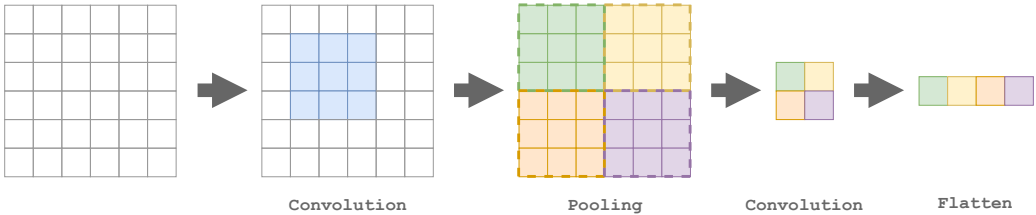
Combining known concepts to represent new ones.

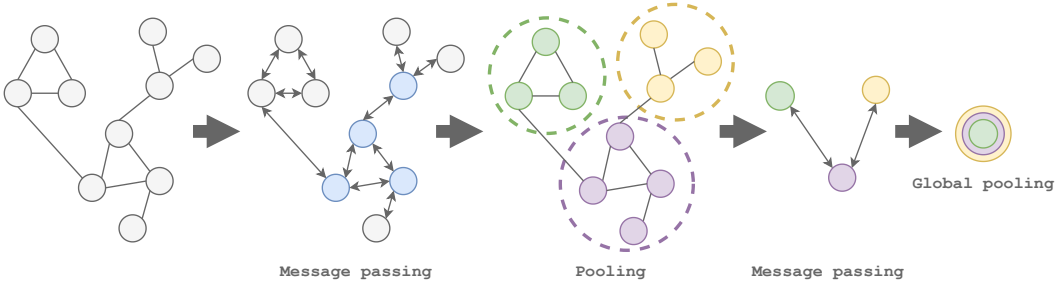
[1] P. W. Battaglia *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

From CNNs to GNNs

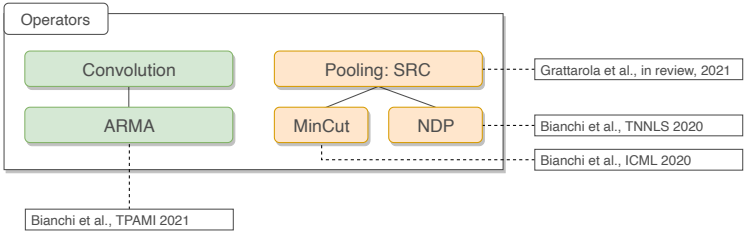


From CNNs to GNNs

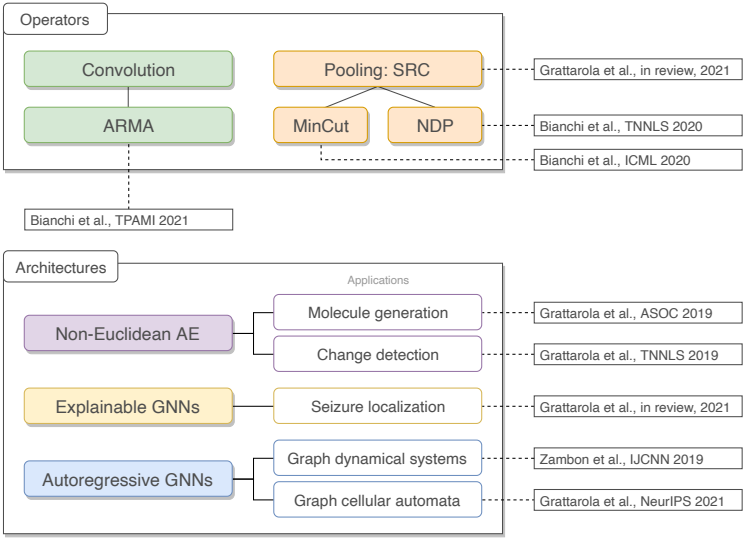




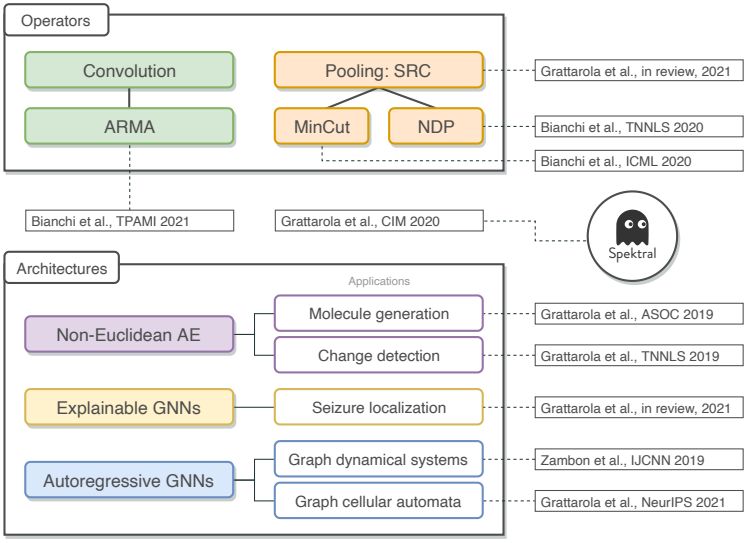
Research summary



Research summary



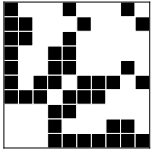
Research summary



Operators

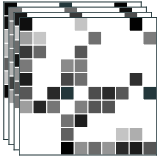
Notation

- **Graph**: nodes connected by edges
- X: node attributes
- E: edge attributes
- A: adjacency matrix
- **Structure operator** S: $s_{ij} \neq 0 \iff a_{ij} \neq 0$

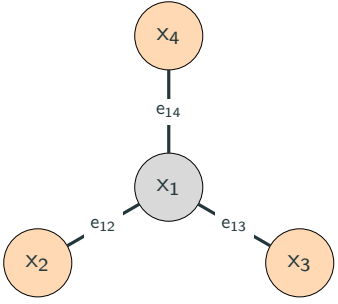


Adjacency matrix
$N \times N$

Node attributes
$N \times D_n$



Edge attributes
$N \times N \times D_e$



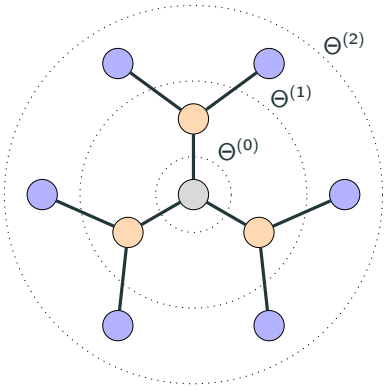
Convolutional operators

Convolutional operators - Polynomial

Typical graph convolutional networks compute a polynomial of the structure operator S [2], [3]:

$$X' = \sum_{k=0}^{K} S^k X \Theta^{(k)}$$

with learnable weights $\Theta^{(k)} \in \mathbb{R}^{D_n \times D'_n}$.



[2] M. Defferrard *et al.*, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in Neural Information Processing Systems*, 2016.

[3] T. N. Kipf *et al.*, "Semi-Supervised Classification with Graph Convolutional Networks," *International Conference on Learning Representations (ICLR)*, 2017.

Convolutional operators - Rational

Convolution with rational filter:

$$X' = \left(I + \sum_{k=1}^{K} q_k S^k\right)^{-1} \left(\sum_{k=0}^{K-1} p_k S^k\right) X$$

[4] E. Isufi *et al.*, "Autoregressive moving average graph filtering," *arXiv preprint arXiv:1602.04436*, 2016.

Convolutional operators - Rational

Convolution with rational filter:

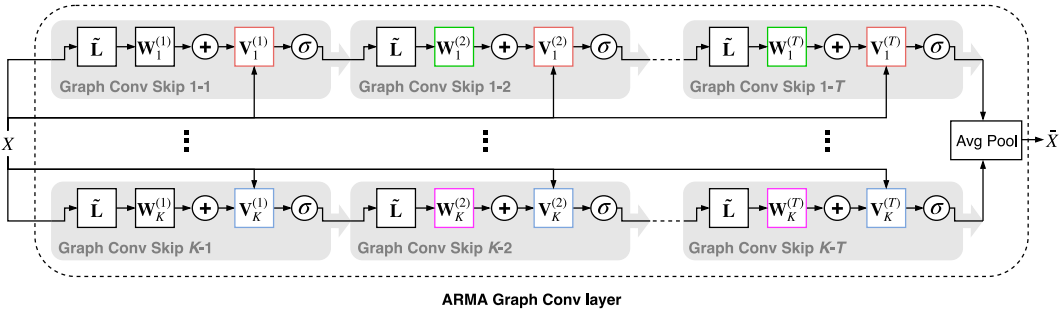
$$X' = \left(I + \sum_{k=1}^{K} q_k S^k\right)^{-1} \left(\sum_{k=0}^{K-1} p_k S^k\right) X$$

ARMA approximation of a rational filter [4]:

$$X^{(t+1)} = aSX^{(t)} + bX$$

[4] E. Isufi *et al.*, "Autoregressive moving average graph filtering," *arXiv preprint arXiv:1602.04436*, 2016.

ARMA GNNs [5]



Approximate recursion with finite number of propagation steps: $\mathrm{X}^{(t+1)} = \sigma\left(\mathrm{S}\mathrm{X}^{(t)}\mathrm{W} + \mathrm{X}\mathrm{V}\right)$

[5] F. M. Bianchi, D. Grattarola, *et al.*, "Graph neural networks with convolutional arma filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

ARMA GNNs - Results

Table 1: Node classification accuracy.

Method	Cora	Citeseer	Pubmed	PPI
GAT	83.1 ±0.6	70.9 ±0.6	78.5 ±0.3	81.3 ±0.1
GraphSAGE	73.7 ±1.8	65.9 ±0.9	78.5 ±0.6	70.0 ±0.0
GIN	75.1 ±1.7	63.1 ±2.0	77.1 ±0.7	78.1 ±2.6
GCN	81.5 ±0.4	70.1 ±0.7	**79.0** ±0.5	80.8 ±0.1
Chebyshev	79.5 ±1.2	70.1 ±0.8	74.4 ±1.1	86.4 ±0.1
CayleyNet	81.2 ±1.2	67.1 ±2.4	75.6 ±3.6	84.9 ±1.2
ARMA	**83.4** ±0.6	**72.5** ±0.4	78.9 ±0.3	**90.5** ±0.3

Table 2: Graph classification accuracy.

Method	Enzymes	Proteins	D&D	MUTAG	BHard
GAT	51.7±4.3	72.3±3.1	70.9±4.0	87.3±5.3	30.1±0.7
GraphSAGE	60.3±7.1	70.2±3.9	73.6±4.1	85.7±4.7	71.8±1.0
GIN	45.7±7.7	71.4±4.5	71.2±5.4	86.3±9.1	72.1±1.1
GCN	53.0±5.3	71.0±2.7	74.7±3.8	85.7±6.6	71.9±1.2
Chebyshev	57.9±2.6	72.1±3.5	73.7±3.7	82.6±5.2	71.3±1.2
CayleyNet	43.1±10.7	65.6±5.7	70.3±11.6	87.8±10.0	70.7±2.4
ARMA	**60.6**±7.2	**73.7**±3.4	**77.6**±2.7	**91.5**±4.2	**74.1**±0.5

Table 3: Graph signal classification accuracy.

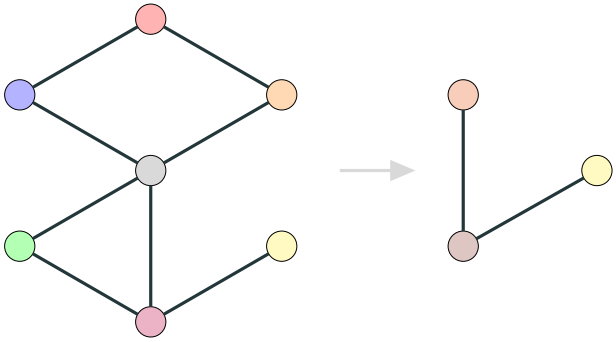
GNN layer	MNIST	20news
GCN	98.48 ± 0.2	65.45 ± 0.2
Chebyshev	99.14 ± 0.1	68.24 ± 0.2
CayleyNet	99.18 ± 0.1	68.84 ± 0.3
ARMA	**99.20** ± 0.1	**70.02** ± 0.1

Table 4: Graph regression mean squared error.

Property	GCN	Chebyshev	CayleyNet	ARMA
mu	0.445±0.007	0.433±0.003	0.442±0.009	**0.394**±0.005
alpha	0.141±0.016	0.171±0.008	0.118±0.005	**0.098**±0.005
HOMO	0.371±0.030	0.391±0.012	0.336±0.007	**0.326**±0.010
LUMO	0.584±0.051	0.528±0.005	0.679±0.148	**0.508**±0.011
gap	0.650±0.070	0.565±0.015	0.758±0.106	**0.552**±0.013
R2	0.132±0.005	0.294±0.022	0.185±0.043	**0.119**±0.019
ZPVE	0.349±0.022	0.358±0.001	0.555±0.174	**0.338**±0.001
U0_atom	0.064±0.003	0.126±0.017	1.493±1.414	**0.053**±0.004
Cv	0.192±0.012	0.215±0.010	0.184±0.009	**0.163**±0.007

Pooling operators

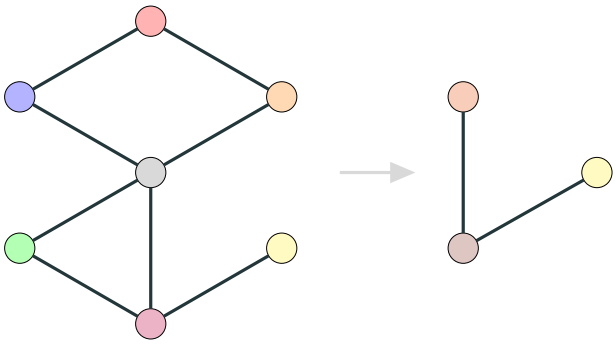
Pooling operators



Goal: reduce the size of the graph.

[6] D. Grattarola *et al.*, "Understanding Pooling in Graph Neural Networks," *Under review at IEEE TNNLS*, 2021.

Pooling operators

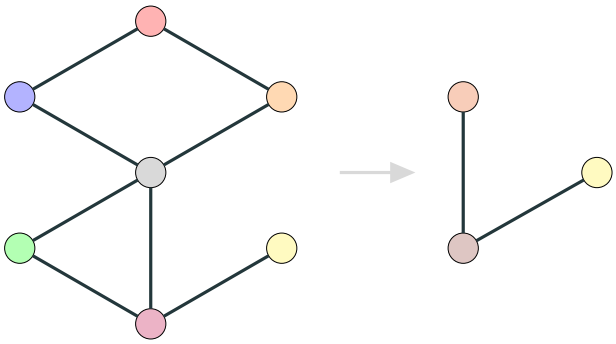


Goal: reduce the size of the graph.
Why:

- Computational cost
- Invariance to feature location
- Abstraction

[6] D. Grattarola *et al.*, "Understanding Pooling in Graph Neural Networks," *Under review at IEEE TNNLS*, 2021.

Pooling operators



Goal: reduce the size of the graph.
Why:

- Computational cost
- Invariance to feature location
- Abstraction

Idea: 3-step process proposed in [6]: *Select*, *Reduce*, *Connect* (SRC).

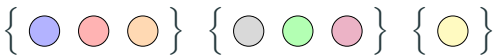
[6] D. Grattarola *et al.*, "Understanding Pooling in Graph Neural Networks," *Under review at IEEE TNNLS*, 2021.

Pooling operators as SRC

Method	Select	Reduce	Connect
DiffPool [7]	$\mathbf{S}^\top = \mathrm{GNN}_1(\mathbf{A}, \mathbf{X})$ (w/ auxiliary loss)	$\mathbf{X}' = \mathbf{S} \cdot \mathrm{GNN}_2(\mathbf{A}, \mathbf{X})$	$\mathbf{A}' = \mathbf{S}\mathbf{A}\mathbf{S}^\top$
MinCut [8]	$\mathbf{S}^\top = \mathrm{MLP}(\mathbf{X})$ (w/ auxiliary loss)	$\mathbf{X}' = \mathbf{S}\mathbf{X}$	$\mathbf{A}' = \mathbf{S}\mathbf{A}\mathbf{S}^\top$
NMF [9]	Factorise: $\mathbf{A} = \mathbf{W}\mathbf{H} \rightarrow \mathbf{S} = \mathbf{H}$	$\mathbf{X}' = \mathbf{S}\mathbf{X}$	$\mathbf{A}' = \mathbf{S}\mathbf{A}\mathbf{S}^\top$
LaPool [10]	$\begin{cases} \mathbf{V} = \lVert \mathbf{L}\mathbf{X} \rVert_d ; \\ \mathbf{i} = \{i \mid \mathbf{V}_i > \mathbf{V}_j, \forall j \in \mathcal{N}(i)\} \\ \mathbf{S}^\top = \mathrm{SparseMax}\left(\beta \frac{\mathbf{X}\mathbf{X}_\mathbf{i}^\top}{\lVert \mathbf{X} \rVert \lVert \mathbf{X}_\mathbf{i} \rVert}\right) \end{cases}$	$\mathbf{X}' = \mathbf{S}\mathbf{X}$	$\mathbf{A}' = \mathbf{S}\mathbf{A}\mathbf{S}^\top$
Graclus [11]	$\mathcal{S}_k = \left\{ \mathbf{x}_i, \mathbf{x}_j \mid \arg\max_j \left(\frac{\mathbf{A}_{ij}}{\mathbf{D}_{ii}} + \frac{\mathbf{A}_{ij}}{\mathbf{D}_{jj}} \right) \right\}$	$\mathbf{X}' = \mathbf{S}\mathbf{X}$	METIS [12]
NDP [13]	$\mathbf{i} = \{i \mid \mathbf{u}_{max,i} > 0\}$	$\mathbf{X}' = \mathbf{X}_\mathbf{i}$	Kron r. [14]
Top-K [15]	$\mathbf{y} = \frac{\mathbf{X}\mathbf{p}}{\lVert \mathbf{p} \rVert}$; $\mathbf{i} = \mathrm{top}_K(\mathbf{y})$	$\mathbf{X}' = (\mathbf{X} \odot \sigma(\mathbf{y}))_\mathbf{i}$;	$\mathbf{A}' = \mathbf{A}_{\mathbf{i},\mathbf{i}}$
SAGPool [16]	$\mathbf{y} = \mathrm{GNN}(\mathbf{A}, \mathbf{X})$; $\mathbf{i} = \mathrm{top}_K(\mathbf{y})$	$\mathbf{X}' = (\mathbf{X} \odot \sigma(\mathbf{y}))_\mathbf{i}$;	$\mathbf{A}' = \mathbf{A}_{\mathbf{i},\mathbf{i}}$

Selection

The **selection** stage computes K **supernodes**:



$$\text{Sel} : \mathcal{G} \mapsto \mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_K\},$$

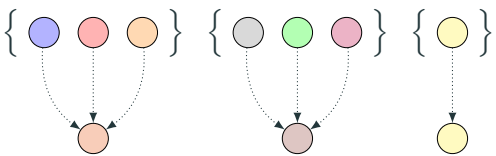
$$\mathcal{S}_k = \{(\mathsf{x}_i, \mathsf{s}_i) \mid \mathsf{s}_i \in \mathbb{R}_{>0}\}\,.$$

$$\mathsf{S} = \quad \in \mathbb{R}^{K \times N}$$

Reduction

The **reduction** stage aggregates the supernodes in a **permutation-invariant** way:

$$\text{Red} : \mathcal{G}, \mathcal{S}_k \mapsto \mathrm{x}'_k$$



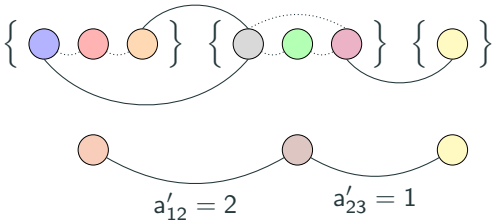
Typical approach: $\mathrm{X}' = \mathrm{S}\mathrm{X}$ $(\in \mathbb{R}^{K \times F})$

Connection

The **connection** function decides whether two supernodes are connected:

$$\text{Con} : \mathcal{G}, \mathcal{S}_k, \mathcal{S}_l \mapsto e'_{kl}$$

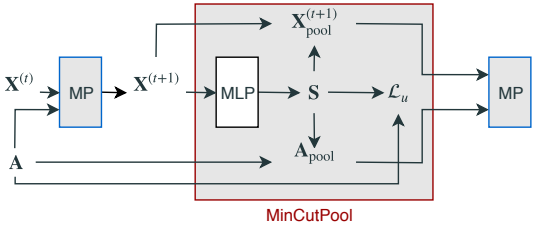
Typical approach: $A' = SAS^\top \quad (\in \mathbb{R}^{K \times K})$



MinCut pooling [8]

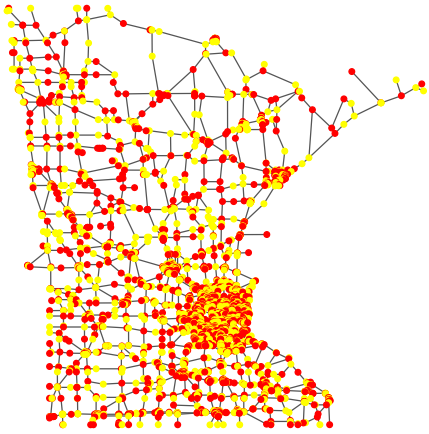
- Select: $\mathbf{S}^\top = \text{MLP}(\mathbf{X})$
- Reduce: $\mathbf{X}' = \mathbf{SX}$
- Connect: $\mathbf{A}' = \mathbf{SAS}^\top$
- Custom loss:

$$\mathcal{L} = \underbrace{-\frac{Tr(\mathbf{SAS}^\top)}{Tr(\mathbf{SDS}^\top)}}_{\text{MinCut}} + \underbrace{\left\| \frac{\mathbf{SS}^\top}{\|\mathbf{SS}^\top\|_F} - \frac{\mathbf{I}_K}{\sqrt{K}} \right\|_F}_{\text{Orthogonality}}$$



[8] F. M. Bianchi, D. Grattarola, *et al.*, "Spectral Clustering with Graph Neural Networks for Graph Pooling," *International Conference on Machine Learning*, 2020.

Node decimation pooling [13]



- **Idea**: regular subsampling of the nodes using highest-frequency eigenvector $\mathrm{u}_{\max}$.
- Select: $\mathrm{I} = \{i \mid \mathrm{u}_{\max}[i] > 0\}$
- Reduce: $\mathrm{X}' = \mathrm{X}_{\mathrm{I}}$
- Connect: Kron reduction [14]

[14] F. Dorfler *et al.*, "Kron Reduction of Graphs With Applications to Electrical Networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 1, 2013.

[13] F. M. Bianchi, D. Grattarola, *et al.*, "Hierarchical representation learning in graph neural networks with node decimation pooling," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Pooling - Results

Table 5: Graph classification accuracy.

Dataset	WL	Dense	No-pool	Graclus	NDP	DiffPool	Top-K	SAGpool	MinCut
Bench-easy	92.6±0.0	29.3±0.3	98.5±0.3	97.5±0.5	97.9±0.5	98.6±0.4	82.4±8.9	84.2±2.3	**99.0**±0.0
Bench-hard	60.0±0.0	29.4±0.3	67.6±2.8	69.0±1.5	**72.6**±0.9	69.9±1.9	42.7±15.2	37.7±14.5	**73.8**±1.9
Mutagen.	**81.7**±1.1	68.4±0.3	78.0±1.3	74.4±1.8	77.8±2.3	77.6±2.7	71.9±3.7	72.4±2.4	79.9±2.1
Proteins	71.2±2.6	68.7±3.3	72.6±4.8	68.6±4.6	73.3±3.7	72.7±3.8	69.6±3.5	70.5±2.6	**76.5**±2.6
DD	78.6±2.7	70.6±5.2	76.8±1.5	70.5±4.8	72.0±3.1	**79.3**±2.4	69.4±7.8	71.5±4.5	**80.8**±2.3
COLLAB	74.8±1.3	79.3±1.6	**82.1**±1.8	77.1±2.1	79.1±1.5	81.8±1.4	79.3±1.8	79.2±2.0	**83.4**±1.7
Reddit-B	68.2±1.7	48.5±2.6	80.3±2.6	79.2±0.4	84.3±2.4	86.8±2.1	74.7±4.5	73.9±5.1	**91.4**±1.5

MinCut achieves state of the art on many graph classification benchmarks (results from [8]).

[8] F. M. Bianchi, D. Grattarola, *et al.*, "Spectral Clustering with Graph Neural Networks for Graph Pooling," *International Conference on Machine Learning*, 2020.

Pooling operators - Results

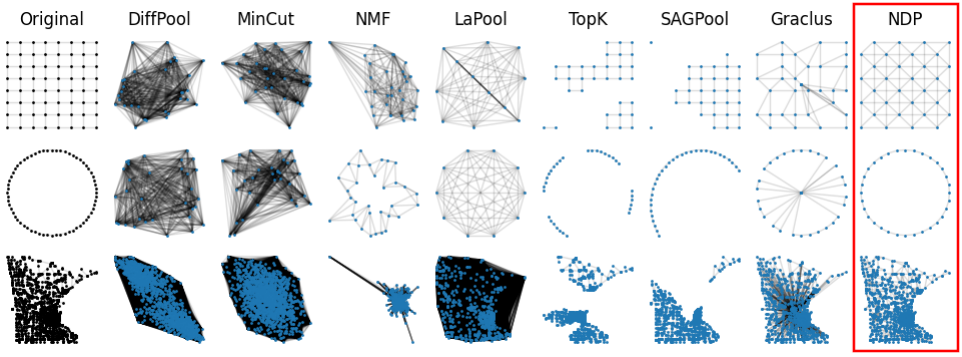
Table 6: Reconstruction MSE (scale of 10^{-3}).

	DiffPool	MinCut	NMF	LaPool	TopK	SAGPool	Graclus	NDP
Grid2d	0.010 ±0.005	0.002 ±0.002	**0.000** ±0.000	0.002 ±0.001	18.86 ±3.923	16.61 ±3.270	0.109 ±0.000	**0.000** ±0.000
Ring	0.018 ±0.003	0.001 ±0.000	**0.000** ±0.000	0.052 ±0.046	132.2 ±4.133	148.5 ±30.10	0.600 ±0.000	**0.000** ±0.000
Bunny	3.901 ±0.275	**0.208** ±0.034	0.339 ±0.055	0.610 ±0.103	15.32 ±3.557	16.10 ±1.722	0.332 ±0.043	0.373 ±0.070
Airplane	0.094 ±0.022	0.005 ±0.002	0.020 ±0.000	**0.002** ±0.000	0.096 ±0.028	0.268 ±0.081	0.009 ±0.000	0.012 ±0.000
Car	0.143 ±0.127	0.535 ±0.200	0.016 ±0.001	OOR	0.229 ±0.023	0.204 ±0.029	0.102 ±0.000	**0.009** ±0.000
Guitar	0.101 ±0.025	0.313 ±0.000	0.007 ±0.000	OOR	0.056 ±0.051	0.060 ±0.044	0.010 ±0.000	**0.005** ±0.000
Person	0.077 ±0.041	0.301 ±0.000	**0.001** ±0.000	OOR	0.055 ±0.012	0.062 ±0.033	**0.001** ±0.000	**0.001** ±0.000

NDP works better for geometric data, where regular subsampling is desirable (results from [6]).

[6] D. Grattarola *et al.*, "Understanding Pooling in Graph Neural Networks," *Under review at IEEE TNNLS*, 2021.

Pooling operators - Results



NDP works better for geometric data, where regular subsampling is desirable (results from [6]).

[6] D. Grattarola *et al.*, "Understanding Pooling in Graph Neural Networks," *Under review at IEEE TNNLS*, 2021.

Architectures

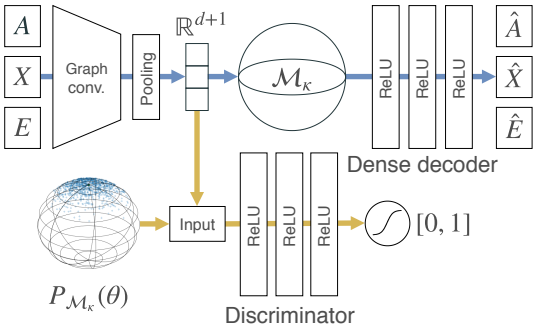
Architectures

- Non-Euclidean Autoencoders
- Explainable GNNs
- Auto-regressive GNNs

Non-Euclidean Autoencoders

Non-Euclidean Autoencoders [17]

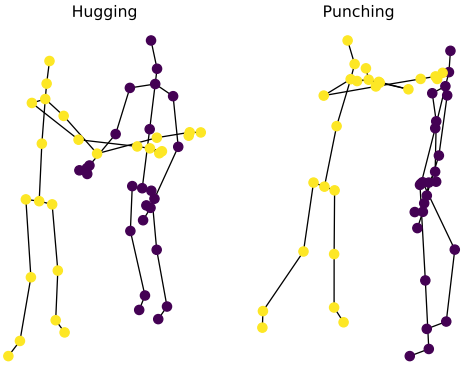
Goal: represent graphs on a non-Euclidean manifold with constant curvature (CCM).
How: adversarial AE with non-Euclidean prior.

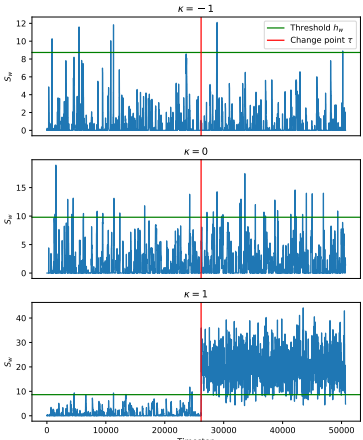


$$\min_{f_{\text{enc}}} \max_{f_{\text{dis}}} \mathbb{E}_{z\sim p(z)} \left[\log f_{\text{dis}}(z)\right] + \mathbb{E}_{x\sim p_{\text{data}}(x)} \left[\log(1 - (f_{\text{dis}} \circ f_{\text{enc}})(x)\right].$$

[17] D. Grattarola *et al.*, "Adversarial autoencoders with constant-curvature latent manifolds," *Applied Soft Computing*, 2019.

Change detection on CCMs [18]

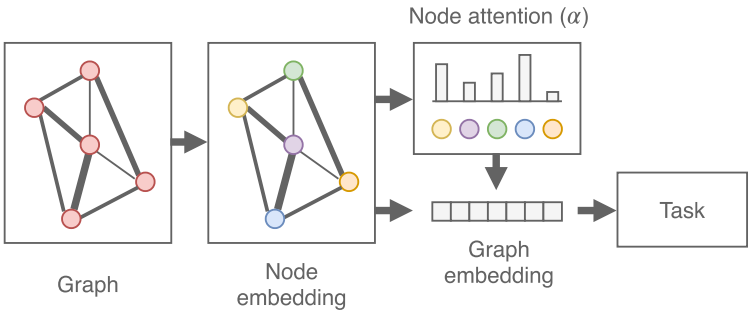




[18] D. Grattarola *et al.*, "Change Detection in Graph Streams by Learning Graph Embeddings on Constant-Curvature Manifolds," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

Explainable GNNs

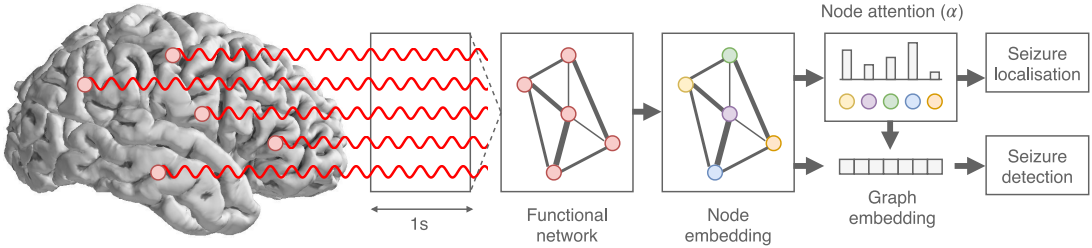
Explainable GNNs



Make GNN explainable by introducing attention in the readout operation:

$$\mathsf{z} = \text{Attn-RO}(\mathsf{h}) = \sum_{j=1}^{N} \alpha_j \mathsf{h}_j, \text{ where } \alpha_j = \frac{\exp\left(\mathsf{h}_j \cdot \mathsf{a}\right)}{\sum_{k=1}^{N} \exp\left(\mathsf{h}_k \cdot \mathsf{a}\right)},$$

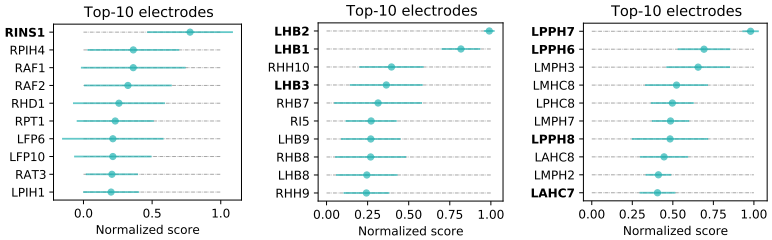
Explainable GNNs for seizure localization [19]



Idea: use attention scores to identify the brain areas where seizures originate.

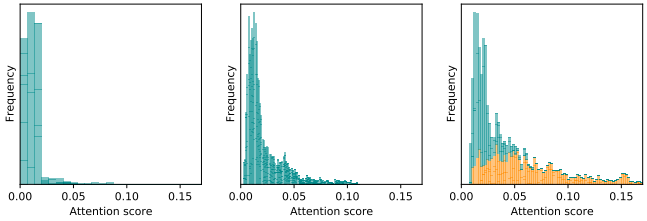
[19] D. Grattarola *et al.*, "Unsupervised seizure localisation with attention-based graph neural networks," *Under review at IEEE TBME*, 2021.

Seizure localization - Results



Known seizure onset zone: GNN shows strong correlation with clinical diagnosis.

Seizure localization - Results



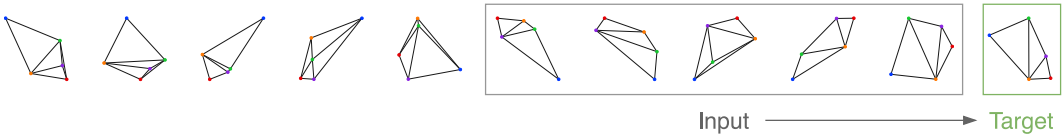
(a) Unknown SOZ **(b)** Unknown SOZ **(c)** Known SOZ

Unknown seizure onset zone: GNN also shows uncertainty.

Auto-regressive GNNs

Auto-regressive GNNs [20]

Goal: predict the next observation of a graph-valued dynamical system.



[20] D. Zambon *et al.*, "Autoregressive Models for Sequences of Graphs," *International Joint Conference on Neural Networks*, 2019.

Autoregressive GNNs

Traditional AR model

$$f : \mathbb{R}^{p \times d} \to \mathbb{R}^d$$

$$x_{t+1} = f(x_t, x_{t-1}, \ldots, x_{t-p+1}) + \epsilon$$

$$\mathbb{E}[\epsilon_i] = 0$$

$$Var[\epsilon_i] = \sigma^2 < \infty$$

Graph AR model

Autoregressive GNNs

Traditional AR model

$$f : \mathbb{R}^{p \times d} \to \mathbb{R}^d$$

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-p+1}) + \epsilon$$

$$\mathbb{E}[\epsilon_i] = 0$$

$$Var[\epsilon_i] = \sigma^2 < \infty$$

Graph AR model

$$\phi : \mathfrak{G}^p \to \mathfrak{G}$$

Autoregressive GNNs

Traditional AR model

$$f : \mathbb{R}^{p \times d} \to \mathbb{R}^d$$

$$x_{t+1} = f(x_t, x_{t-1}, \ldots, x_{t-p+1}) + \epsilon$$

$$\mathbb{E}[\epsilon_i] = 0$$

$$Var[\epsilon_i] = \sigma^2 < \infty$$

Graph AR model

$$\phi : \mathfrak{G}^p \to \mathfrak{G}$$

$$\mathcal{G}_{t+1} = H(\phi(\mathcal{G}_t, \ldots, \mathcal{G}_{t-p+1}), \eta)$$

Autoregressive GNNs

Traditional AR model

$$f : \mathbb{R}^{p \times d} \to \mathbb{R}^d$$

$$x_{t+1} = f(x_t, x_{t-1}, \ldots, x_{t-p+1}) + \epsilon$$

$$\mathbb{E}[\epsilon_i] = 0$$

$$Var[\epsilon_i] = \sigma^2 < \infty$$

Graph AR model

$$\phi : \mathfrak{G}^p \to \mathfrak{G}$$

$$\mathcal{G}_{t+1} = H(\phi(\mathcal{G}_t, \ldots, \mathcal{G}_{t-p+1}), \eta)$$

$$\phi(\mathcal{G}_t, \ldots, \mathcal{G}_{t-p+1}) \in \mathbb{E}^{\mathrm{f}}_{\eta}[H(\phi(\mathcal{G}_t, \ldots, \mathcal{G}_{t-p+1}), \eta)]$$

Where:

$$\mathbb{E}^{\mathrm{f}}[\mathcal{G}] = \arg\min_{\mathcal{G}' \in \mathfrak{G}} \int_{\mathfrak{G}} d(\mathcal{G}, \mathcal{G}')^2 \, dQ(\mathcal{G})$$

Autoregressive GNNs

Traditional AR model	**Graph AR model**
$f : \mathbb{R}^{p \times d} \to \mathbb{R}^d$	$\phi : \mathfrak{G}^p \to \mathfrak{G}$
$x_{t+1} = f(x_t, x_{t-1}, \ldots, x_{t-p+1}) + \epsilon$	$\mathcal{G}_{t+1} = H(\phi(\mathcal{G}_t, \ldots, \mathcal{G}_{t-p+1}), \eta)$
$\mathbb{E}[\epsilon_i] = 0$	$\phi(\mathcal{G}_t, \ldots, \mathcal{G}_{t-p+1}) \in \mathbb{E}^{\mathrm{f}}_{\eta}[H(\phi(\mathcal{G}_t, \ldots, \mathcal{G}_{t-p+1}), \eta)]$
$Var[\epsilon_i] = \sigma^2 < \infty$	$Var^{\mathrm{f}}[\eta] < \infty$

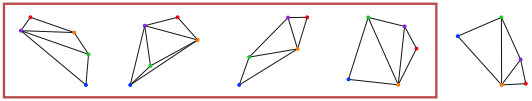
Where:

$$\mathbb{E}^{\mathrm{f}}[\mathcal{G}] = \arg\min_{\mathcal{G}' \in \mathfrak{G}} \int_{\mathfrak{G}} d(\mathcal{G}, \mathcal{G}')^2 \, dQ(\mathcal{G}) \qquad Var^{\mathrm{f}}[\mathcal{G}] := \min_{\mathcal{G}' \in \mathfrak{G}} \int_{\mathfrak{G}} d(\mathcal{G}, \mathcal{G}')^2 \, dQ(\mathcal{G})$$

Auto-regressive GNN

Input

Window of observations up to time t.



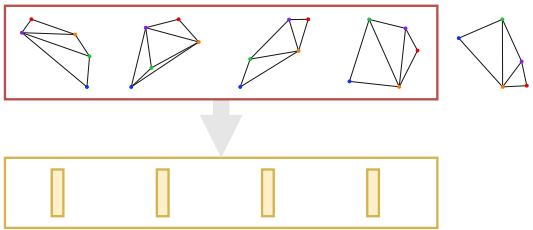
Auto-regressive GNN

Input

Window of observations up to time t.

Graph embedding

GNN block applied in parallel to each graph.



Auto-regressive GNN

Input

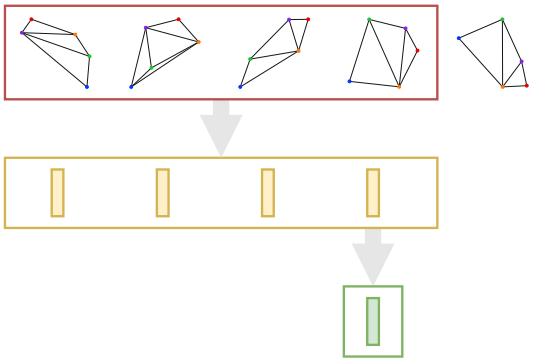
Window of observations up to time t.

Graph embedding

GNN block applied in parallel to each graph.

Predictor

RNN predicts the graph embedding at $t + 1$.



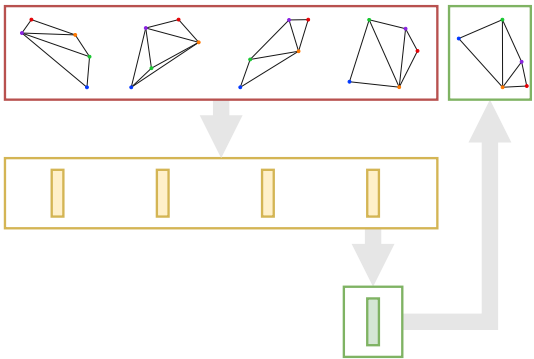
Auto-regressive GNN

Input
Window of observations up to time t.

Graph embedding
GNN block applied in parallel to each graph.

Predictor
RNN predicts the graph embedding at $t+1$.

Graph decoder
Maps the predicted embedding to the predicted graph.



Auto-regressive GNNs - Results

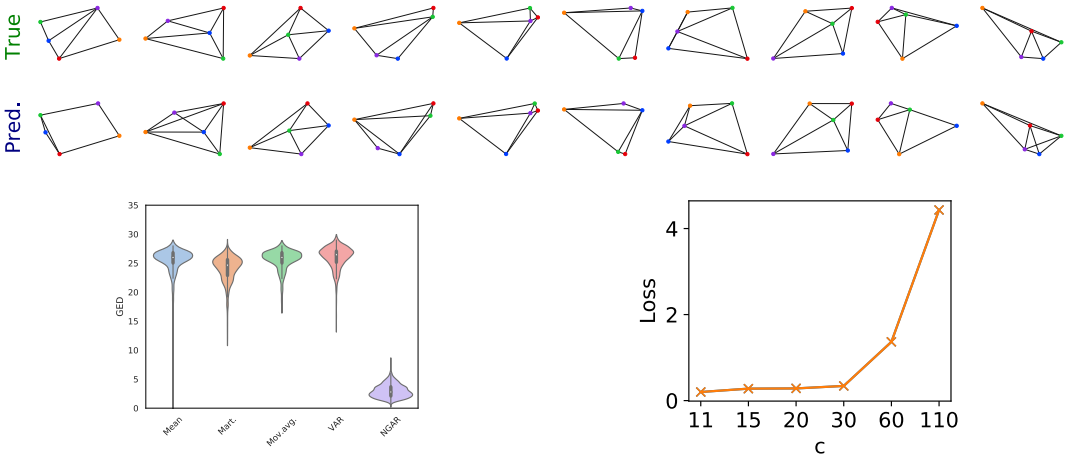


Figure 4: Distance b/w predicted and true graphs.

Figure 5: Loss v. complexity of the problem.

Spektral

Spektral

Python library for GNNs:

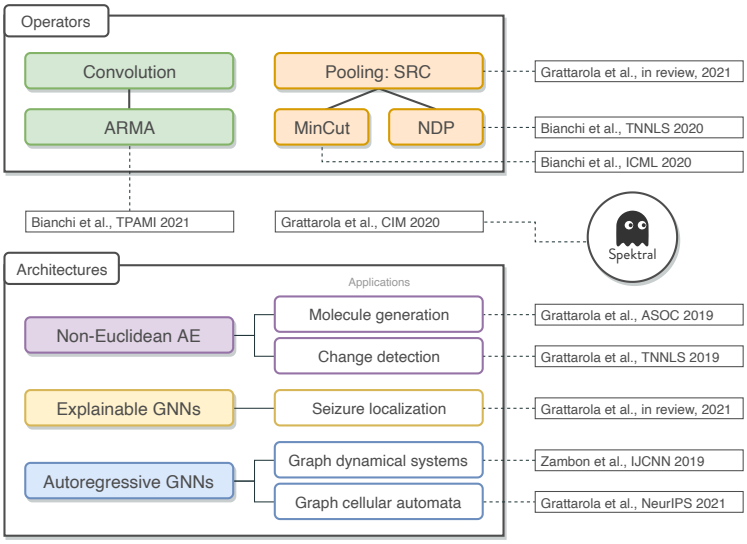
- TensorFlow/Keras
- 25+ layers for convolution and pooling
- Easy to use, flexible, fast
- Almost 2000 ★ on `github.com`

Website: `graphneural.network`
Featured on IEEE Computational Intelligence Magazine [21].

[21] D. Grattarola *et al.*, "Graph neural networks in Tensorflow and Keras with Spektral," *IEEE Computational Intelligence Magazine*, 2021.

Conclusions

Summary


Operators
Convolution
ARMA
Pooling: SRC
MinCut
NDP
Grattarola et al., in review, 2021
Bianchi et al., TNNLS 2020
Bianchi et al., ICML 2020
Bianchi et al., TPAMI 2021
Grattarola et al., CIM 2020
Spektral
Architectures
Applications
Non-Euclidean AE
Molecule generation
Change detection
Grattarola et al., ASOC 2019
Grattarola et al., TNNLS 2019
Explainable GNNs
Seizure localization
Grattarola et al., in review, 2021
Autoregressive GNNs
Graph dynamical systems
Graph cellular automata
Zambon et al., IJCNN 2019
Grattarola et al., NeurIPS 2021

References i

[1] P. W. Battaglia, J. B. Hamrick, V. Bapst, *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[2] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations (ICLR)*, 2017.

[4] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive moving average graph filtering," *arXiv preprint arXiv:1602.04436*, 2016.

[5] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, "Graph neural networks with convolutional arma filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

References ii

[6] D. Grattarola, D. Zambon, F. M. Bianchi, and C. Alippi, "Understanding pooling in graph neural networks," *Under review at IEEE TNNLS*, 2021.

[7] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *arXiv preprint arXiv:1806.08804*, 2018.

[8] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," *International Conference on Machine Learning*, 2020.

[9] D. Bacciu and L. Di Sotto, "A non-negative factorization approach to node pooling in graph convolutional neural networks," in *Proceedings of the 18th International Conference of the Italian Association for Artificial Intelligence*, AIIA, 2019.

[10] E. Noutahi, D. Beani, J. Horwood, and P. Tossou, "Towards interpretable sparse graph representation learning with laplacian pooling," *arXiv preprint arXiv:1905.11577*, 2019.

References iii

[11] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 11, pp. 1944–1957, 2007.

[12] G. Karypis, "Metis: Unstructured graph partitioning and sparse matrix ordering system," *Technical report*, 1997.

[13] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, "Hierarchical representation learning in graph neural networks with node decimation pooling," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[14] F. Dorfler and F. Bullo, "Kron reduction of graphs with applications to electrical networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 1, pp. 150–163, 2013, issn: 1549-8328. doi: 10.1109/TCSI.2012.2215780.

[15] S. J. Hongyang Gao, "Graph u-net," *Submitted to ICLR*, 2019.

References iv

[16] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," *arXiv preprint arXiv:1904.08082*, 2019.

[17] D. Grattarola, L. Livi, and C. Alippi, "Adversarial autoencoders with constant-curvature latent manifolds," *Applied Soft Computing*, 2019.

[18] D. Grattarola, D. Zambon, C. Alippi, and L. Livi, "Change detection in graph streams by learning graph embeddings on constant-curvature manifolds," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[19] D. Grattarola, L. Livi, C. Alippi, R. Wennberg, and T. Valiante, "Unsupervised seizure localisation with attention-based graph neural networks," *Under review at IEEE TBME*, 2021.

[20] D. Zambon, D. Grattarola, L. Livi, and C. Alippi, "Autoregressive models for sequences of graphs," *International Joint Conference on Neural Networks*, 2019.

[21] D. Grattarola and C. Alippi, "Graph neural networks in tensorflow and keras with spektral," *IEEE Computational Intelligence Magazine*, 2021.

[22] A. Sperduti and A. Starita, "Supervised neural networks for the classification of structures," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 714–735, 1997.

[23] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, IEEE, vol. 2, 2005, pp. 729–734.

[24] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[25] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[26] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[27] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212*, 2017.

[28] M. Simonovsky and N. Komodakis, "Dynamic edgeconditioned filters in convolutional neural networks on graphs," in *Proc. CVPR*, 2017.

[29] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Departmental Papers (CIS)*, p. 107, 2000.

[30] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[31] L. Palagi, V. Piccialli, F. Rendl, G. Rinaldi, and A. Wiegele, "Computational approaches to max-cut," in *Handbook on semidefinite, conic and polynomial optimization*, Springer, 2012, pp. 821–847.

[32] C. Bodnar, C. Cangea, and P. Liò, "Deep graph mapper: Seeing graphs through the neural lens," *arXiv preprint arXiv:2002.03864*, 2020.

[33] G. Singh, F. Mémoli, and G. E. Carlsson, "Topological methods for the analysis of high dimensional data sets and 3d object recognition.," in *SPBG*, 2007, pp. 91–100.

Backup

Convolutional operators

History

- **1997**: first GNN (works only for DAGs) [22]
- **2005**: first use of the term "GNN" [23]
- **2009**: improved version of 2005 paper [24]
- **2013**: First GCN [25]
- **2016**: Geometric Deep Learning [26]

[22] A. Sperduti *et al.*, "Supervised neural networks for the classification of structures," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, 1997.

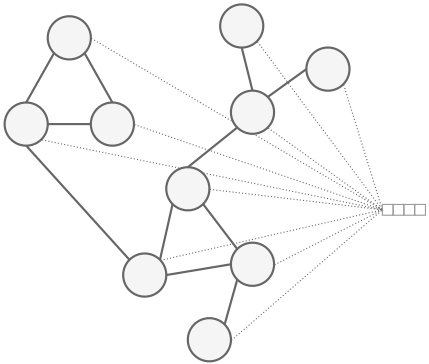
[23] M. Gori *et al.*, "A new model for learning in graph domains," vol. 2, 2005.

[24] F. Scarselli *et al.*, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, 2009.

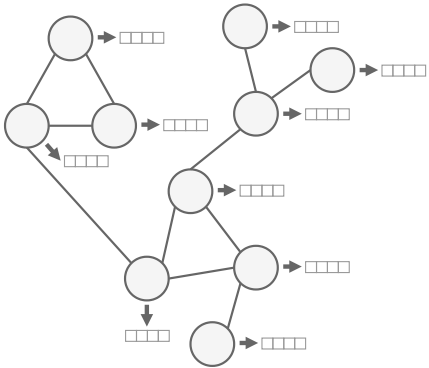
[25] J. Bruna *et al.*, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[26] M. M. Bronstein *et al.*, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, 2017.

Graph-level v. node-level



Graph-level learning.
(e.g., molecules)



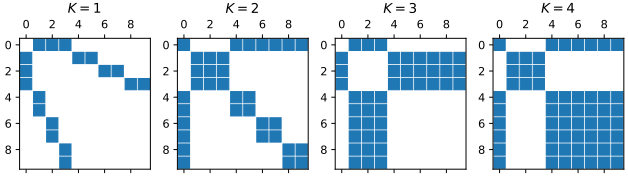
Node-level learning.
(e.g., social networks)

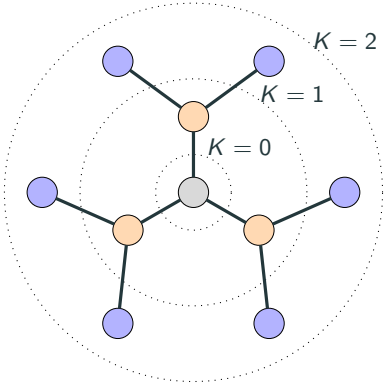
Powers of R

Let's consider the effect of applying R^2 to X:

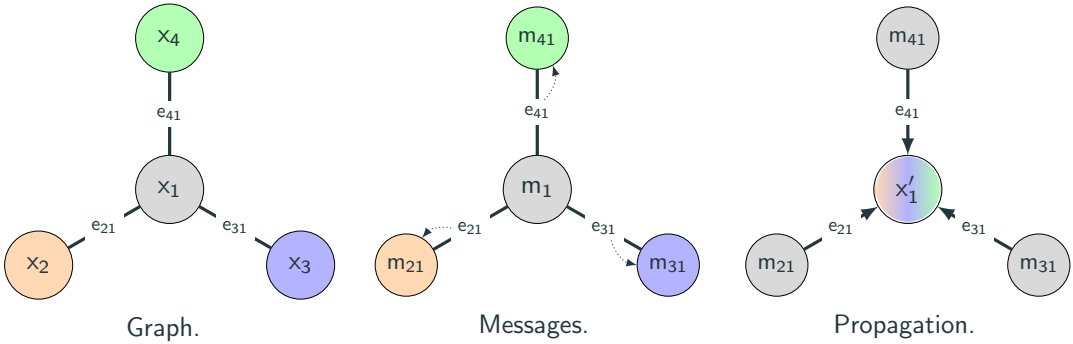
$$(RRX)_i = \sum_{j \in \mathcal{N}(i)} r_{ij} (RX)_j = \sum_{j \in \mathcal{N}(i)} \sum_{k \in \mathcal{N}(j)} r_{ij} \cdot r_{jk} \cdot x_k$$

Key idea: by applying R^K we **read from the K-th order neighbourhood** of a node.





Message Passing Neural Networks [27]



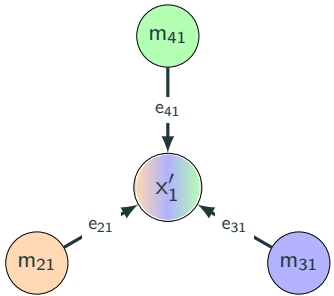
[27] J. Gilmer *et al.*, "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212*, 2017.

Message Passing Neural Networks [27]

A general scheme for message-passing networks:

$$\mathbf{x}_i' = \gamma\left(\mathbf{x}_i, \square_{j \in \mathcal{N}(i)}\, \phi\left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{ji}\right)\right),$$

- ϕ: **message function**, depends on $\mathbf{x}_i$, $\mathbf{x}_j$ and possibly the edge attribute $\mathbf{e}_{ji}$ (we call messages $\mathbf{m}_{ji}$);
- $\square_{j \in \mathcal{N}(i)}$: **aggregation function** (sum, average, max, or something else...);
- γ: **update function**, final transformation to obtain new attributes after aggregating messages.



[27] J. Gilmer *et al.*, "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212*, 2017.

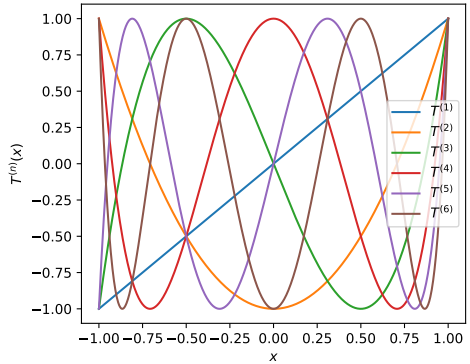
Chebyshev Polynomials [2]

A recursive definition using Chebyshev polynomials:

$$
\begin{aligned}
\mathsf{T}^{(0)} &= \mathsf{I} \\
\mathsf{T}^{(1)} &= \tilde{\mathsf{L}} \\
\mathsf{T}^{(k)} &= 2 \cdot \tilde{\mathsf{L}} \cdot \mathsf{T}^{(k-1)} - \mathsf{T}^{(k-2)}
\end{aligned}
$$

Where $\tilde{\mathsf{L}} = \dfrac{2\mathsf{L}_n}{\lambda_{\max}} - \mathsf{I}$ and $\mathsf{L}_n = \mathsf{I} - \mathsf{D}^{-1/2}\mathsf{A}\mathsf{D}^{-1/2}$

Layer: $\mathsf{X}' = \sigma\Big(\sum_{k=0}^{K} \mathsf{T}^{(k)} \mathsf{X} \Theta^{(k)} \Big)$



[2] M. Defferrard *et al.*, "Convolutional neural networks on graphs with fast localized spectral filtering," ***Advances in Neural Information Processing Systems***, 2016.

Graph Convolutional Networks [3]

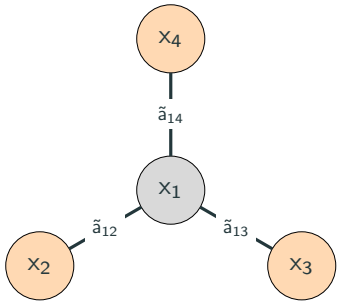
Polynomial of order $K \rightarrow K$ layers of order 1;

Three simplifications:

1. $\lambda_{\max} = 2 \rightarrow \tilde{\mathsf{L}} = \dfrac{2\mathsf{L}_n}{\lambda_{\max}} - \mathsf{I} = -\mathsf{D}^{-1/2}\mathsf{A}\mathsf{D}^{-1/2} = -\mathsf{A}_n$
2. $K = 1 \rightarrow \mathsf{X}' = \mathsf{X}\Theta^{(0)} - \mathsf{A}_n\mathsf{X}\Theta^{(1)}$
3. $\Theta = \Theta^{(0)} = -\Theta^{(1)}$

Layer: $\mathsf{X}' = \sigma\Big(\underbrace{(\mathsf{I} + \mathsf{A}_n)}_{\tilde{\mathsf{A}}}\mathsf{X}\Theta\Big) = \sigma\Big(\tilde{\mathsf{A}}\mathsf{X}\Theta\Big)$

For stability: $\tilde{\mathsf{A}} = \mathsf{D}^{-1/2}(\mathsf{I} + \mathsf{A})\mathsf{D}^{-1/2}$



[3] T. N. Kipf *et al.*, "Semi-Supervised Classification with Graph Convolutional Networks," *International Conference on Learning Representations (ICLR)*, 2017.

Edge-Conditioned Convolution [28]

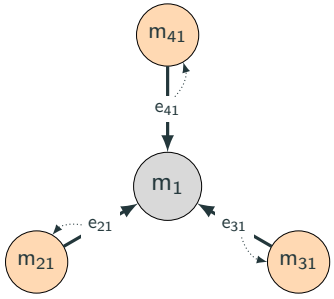
Key idea: incorporate edge attributes into the messages.

Consider a MLP $\phi : \mathbb{R}^S \to \mathbb{R}^{FF'}$ called a **filter generating network**:

$$\Theta^{(ji)} = \text{reshape}(\phi(\mathbf{e}_{ji}))$$

Use the edge-dependent weights to compute messages:

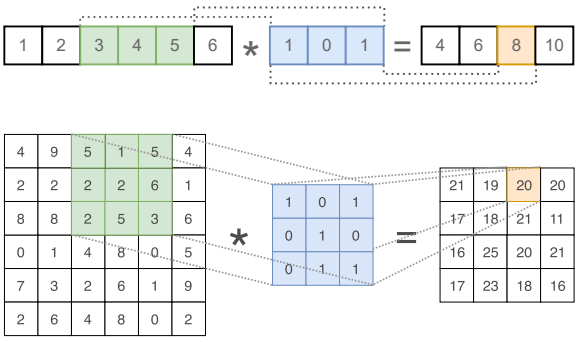
$$\mathbf{x}'_i = \Theta^{(i)}\mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \Theta^{(ji)}\mathbf{x}_j + \mathbf{b}$$



[28] M. Simonovsky *et al.*, "Dynamic edgeconditioned filters in convolutional neural networks on graphs," 2017.

Graph Convolution

Discrete Convolution



Recall: CNNs compute a discrete convolution

$$(f \star g)[n] = \sum_{m=-M}^{M} f[n-m]g[m] \quad (1)$$

Convolution Theorem

Given two functions f and g, their convolution $f \star g$ can be expressed as:

$$f \star g = \mathcal{F}^{-1} \{ \mathcal{F} \{ f \} \cdot \mathcal{F} \{ g \} \} \tag{2}$$

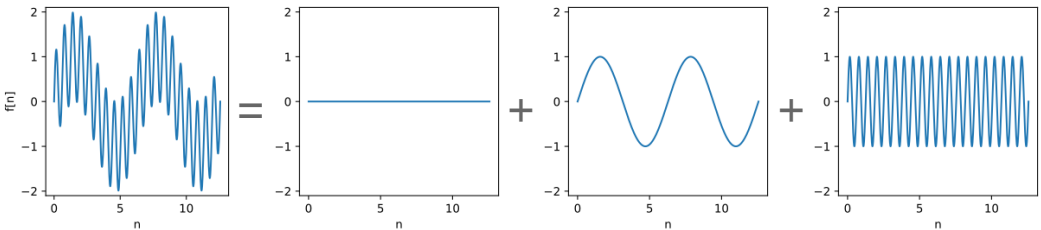
Where $\mathcal{F}$ is the **Fourier transform** and $\mathcal{F}^{-1}$ its inverse.

What is the Fourier transform?

Key intuition – we are representing a function in a different basis.

$$\mathcal{F}\{f\}[k] = \hat{f}[k] = \sum_{n=0}^{N-1} f[n] e^{-i\frac{2\pi}{N}kn}$$

$$\mathcal{F}^{-1}\{\hat{f}\}[n] = f[n] = \frac{1}{N}\sum_{k=0}^{N-1} \hat{f}[k] e^{i\frac{2\pi}{N}kn}$$

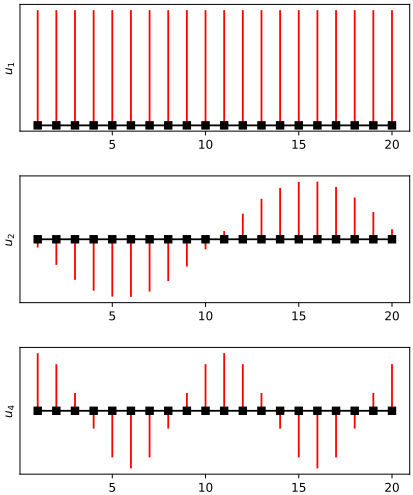


From FT to GFT

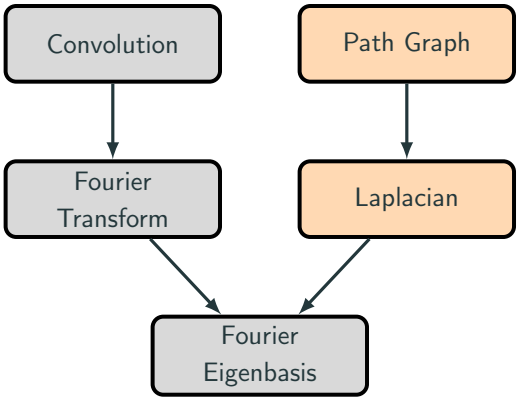
The eigenvectors of the Laplacian for a path graph can be obtained analytically:

$$\mathrm{u}_k[n] = \begin{cases} 1, & \text{for } k = 0 \\ e^{i\pi(k+1)n/N}, & \text{for odd } k, k < N-1 \\ e^{-i\pi kn/N}, & \text{for even } k, k > 0 \\ cos(\pi n), & \text{for odd } k, k = N-1 \end{cases}$$

Looks familiar?



From FT to GFT



- Drop the "grid" assumption
- Replace $e^{-i\frac{2\pi}{N}kn}$ with generic $\mathrm{u}_k[n]$:

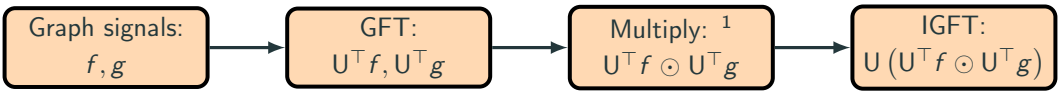
$$\mathcal{F}_G\{f\}[k] = \sum_{n=0}^{N-1} f[n]\mathrm{u}_k[n]$$

- GFT: $\mathcal{F}_G\{f\} = \hat{f} = \mathrm{U}^\top f$;
- IGFT: $\mathcal{F}_G^{-1}\{\hat{f}\} = f = \mathrm{U}\hat{f}$

Graph Convolution

Recall:

- Convolution theorem: $f \star g = \mathcal{F}^{-1}\left\{\mathcal{F}\left\{f\right\} \cdot \mathcal{F}\left\{g\right\}\right\}$
- Spectral theorem: $\mathsf{L} = \mathsf{U}\Lambda\mathsf{U}^\top = \sum_{i=0}^{N-1} \lambda_i \mathsf{u}_i \mathsf{u}_i^\top$

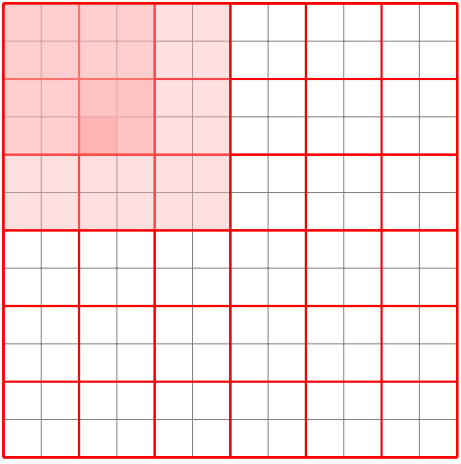


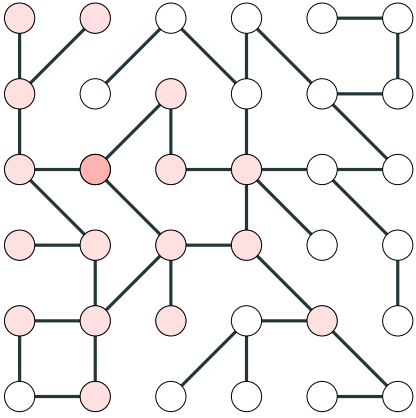
Graph filter: $\mathsf{U}\left(\mathsf{U}^\top f \odot \mathsf{U}^\top g\right) = \mathsf{U} \cdot \underbrace{\mathrm{diag}(\mathsf{U}^\top g)}_{g(\Lambda)} \cdot \mathsf{U}^\top f = \underbrace{\mathsf{U} \cdot g(\Lambda) \cdot \mathsf{U}^\top}_{g(\mathsf{L})} f = g(\mathsf{L}) f$

[1] $\odot$ indicates element-wise multiplication

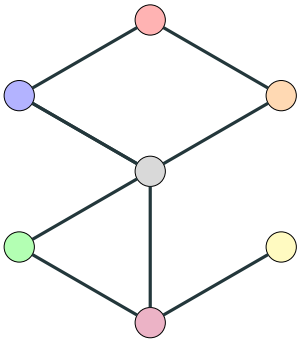
Pooling operators

Pooling in CNNs

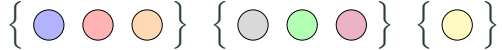




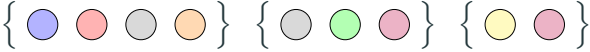
Selecting nodes



Example 1: partition.



Example 2: cover (possible overlaps).



Example 3: sparse.

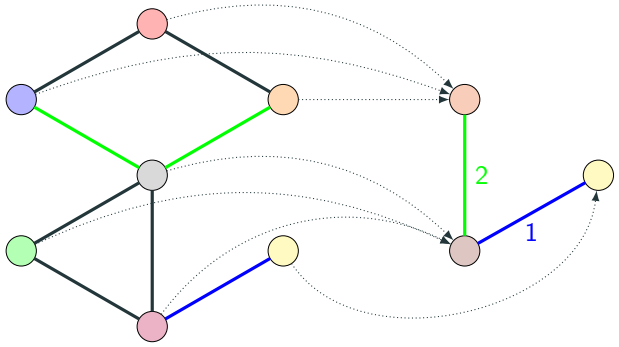
Select, Reduce, Connect [6]

Putting everything together:

$$\underbrace{\mathcal{S} = \{\mathcal{S}_k\}_{k=1:K} = \mathsf{Sel}(\mathcal{G})}_{\text{Selection}};$$

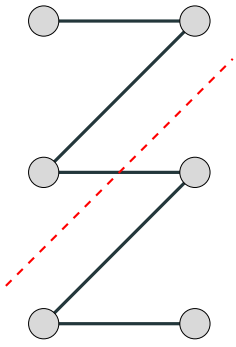
$$\underbrace{\mathcal{X}' = \{\mathsf{Red}(\mathcal{G}, \mathcal{S}_k)\}_{k=1:K}}_{\text{Reduction}};$$

$$\underbrace{\mathcal{E}' = \{\mathsf{Con}(\mathcal{G}, \mathcal{S}_k, \mathcal{S}_l)\}_{k,l=1:K}}_{\text{Connection}};$$

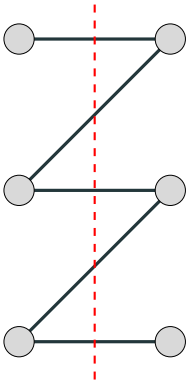


[6] D. Grattarola *et al.*, "Understanding Pooling in Graph Neural Networks," *Under review at IEEE TNNLS*, 2021.

Mincut vs. Maxcut

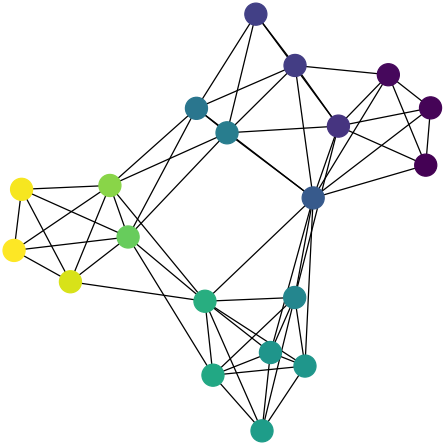


MinCut

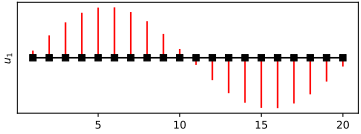


MaxCut

Spectral clustering [30]



The low-frequency eigenvectors naturally cluster the nodes.

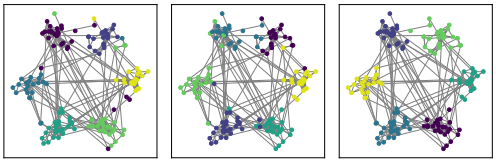


Idea: run k-means clustering (or similar) using the first few eigenvectors.

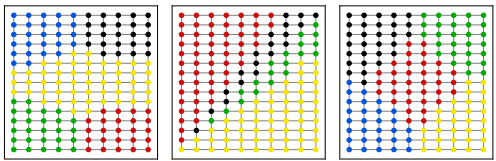
[29] J. Shi *et al.*, "Normalized cuts and image segmentation," *Departmental Papers (CIS)*, 2000.

[30] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, 2007.

MinCut pooling



(a) SC **(b)** DiffPool **(c)** MinCut

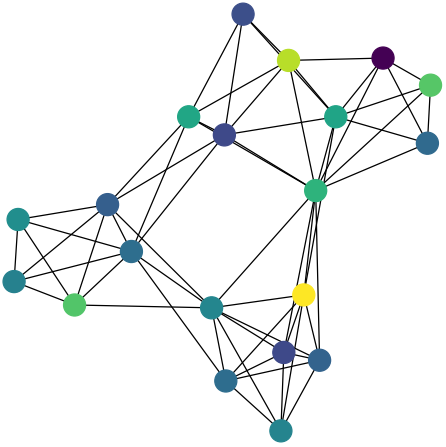


(d) SC **(e)** DiffPool **(f)** MinCut

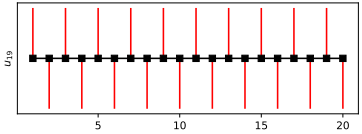
(a) Original **(b)** SC

(c) DiffPool **(d)** MinCut

Node decimation [13]



Alternative: use the highest-frequency eigenvector to do something similar to a regular subsampling.



[31] L. Palagi *et al.*, "Computational approaches to max-cut," 2012.

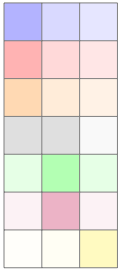
[13] F. M. Bianchi *et al.*, "Hierarchical representation learning in graph neural networks with node decimation pooling," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Learning to pool

Key idea: learn to output $\mathbf{S}^\top$ by giving node features $\mathbf{X}$ as input to a neural network.

- **DiffPool** [7]: GNN for $\mathbf{S}^\top$, regularize with "link prediction" loss;
- **MinCutPool** [8]: MLP for $\mathbf{S}^\top$, regularize with "minimum cut" loss (same objective as spectral clustering);
- **Deep Graph Mapper** [32]: combine Mapper [33] and GCN [3] to compute clusters.

$$\phi(\mathbf{X}) = \mathbf{S}^\top = \quad \in \mathbb{R}^{N \times K}$$

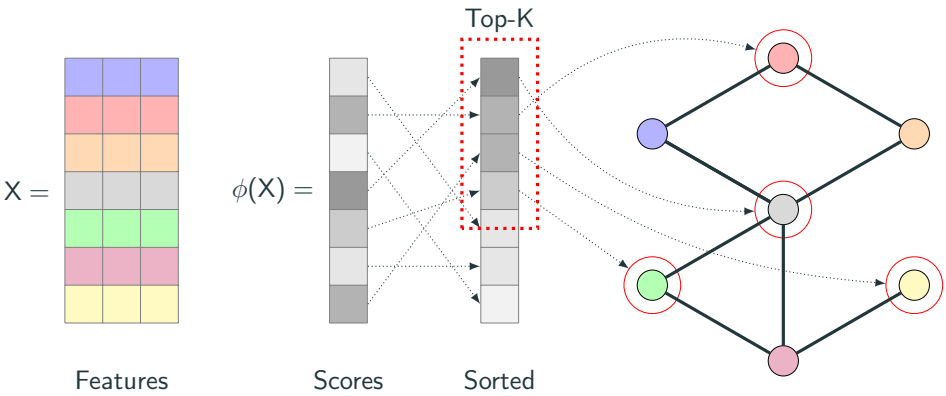


[7] R. Ying *et al.*, "Hierarchical Graph Representation Learning with Differentiable Pooling," *arXiv preprint arXiv:1806.08804*, 2018.

[8] F. M. Bianchi *et al.*, "Spectral Clustering with Graph Neural Networks for Graph Pooling," *International Conference on Machine Learning*, 2020.

[32] C. Bodnar *et al.*, "Deep Graph Mapper: Seeing Graphs through the Neural Lens," *arXiv preprint arXiv:2002.03864*, 2020.

Top-K methods



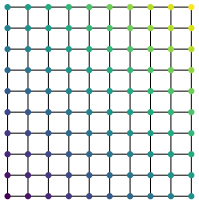
Top-K methods

Reduce: $X' = X_i$ - Connect: $A' = A_{i,i}$

Problems:

- Top-k selection is **non-differentiable** (no way of training ϕ). Solved by **gating** (multiplying) the node attributes with the scores.
- Graph is likely to be **disconnected** or simply **cut off** (like in the image on the right).
  Not really solvable...

Original



Top-K

Main properties of pooling operators

- **Dense vs. Sparse**: how many nodes are selected for the supernodes;
- **Fixed vs. Adaptive**: how many supernodes does the selection compute;
- **Trainable vs. Non-trainable**: learn to pool from data or not;

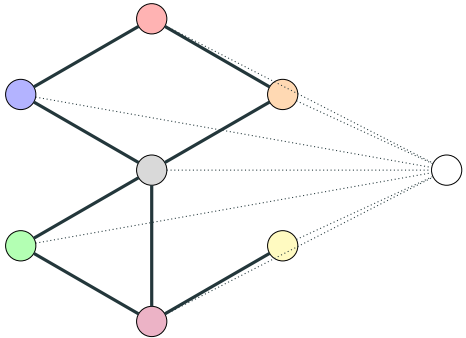
Global Pooling

In CNNs, after convolution, we usually **flatten** out the images to give a vector as input to a MLP:

1	2	3
4	5	6
7	8	9

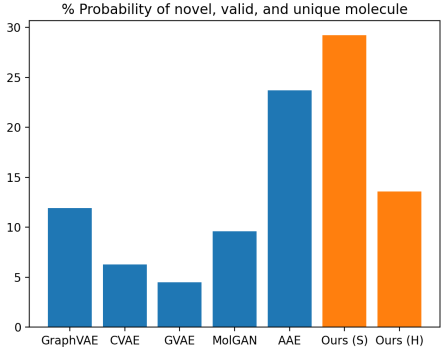
1	2	3	4	5	6	7	8	9

The graph equivalent must be **invariant to permutations** of the nodes:



More results

Molecule generation [17]



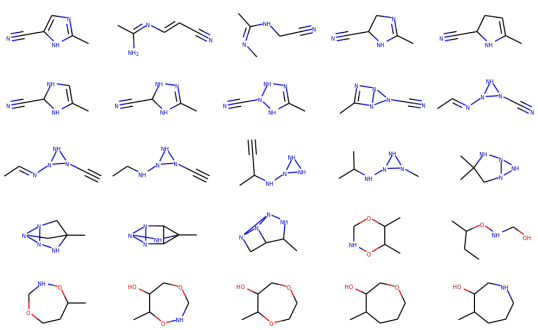


Figure 8: Valid, novel, and unique molecules

[17] D. Grattarola *et al.*, "Adversarial autoencoders with constant-curvature latent manifolds," *Applied Soft Computing*, 2019.