CS6208 : Advanced Topics in Artificial Intelligence
Graph Machine Learning

Lecture 6 : Weisfeiler-Lehman GNNs

Semester 2 2022/23

Xavier Bresson

https://twitter.com/xbresson

Department of Computer Science
National University of Singapore (NUS)

Outline

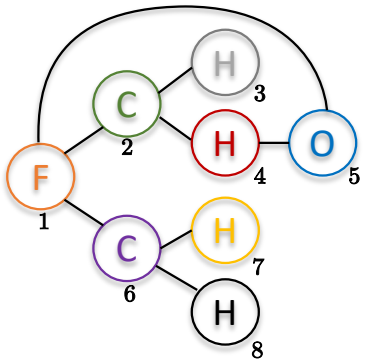
- Weisfeiler-Lehman isomorphism test
- Graph isomorphism networks
- Equivariant graph neural networks
- Expressivity power and generalization
- Equivariant set neural networks
- Conclusion

Outline

- Weisfeiler-Lehman isomorphism test
- Graph isomorphism networks
- Equivariant graph neural networks
- Expressivity power and generalization
- Equivariant set neural networks
- Conclusion

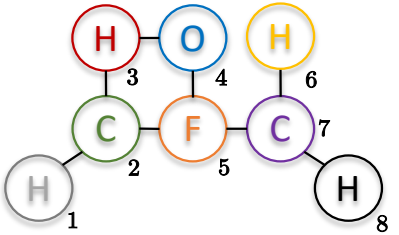
Graph isomorphism

- Graph isomorphism : Two graphs are isomorphic if there exists an index permutation between the nodes that preserves node adjacencies.
  - An example of two isomorphic molecular graphs. Node correspondence is given by the node color.
- Determining whether two graphs are isomorphic is NP-intermediate : It is not known if a polynomial time algorithm exists, or the problem is NP-hard.
- Weisfeiler-Lehman (WL) proposed an algorithm[1] to test if two graphs are not isomorphic.
  - However, the WL test is not sufficient to guarantee that two graphs are isomorphic (necessary condition only).



Molecular graph 1

⇕ Isomorphic graphs



Molecular graph 2

[1] Weisfeiler, Lehman, A reduction of a graph to a canonical form and an algebra arising during this reduction, 1968

Weisfeiler-Lehman test[1]

- Design an injective "coloring" function f_{WL} that takes a pair (node, its neighborhood) as input, and outputs a new node color :

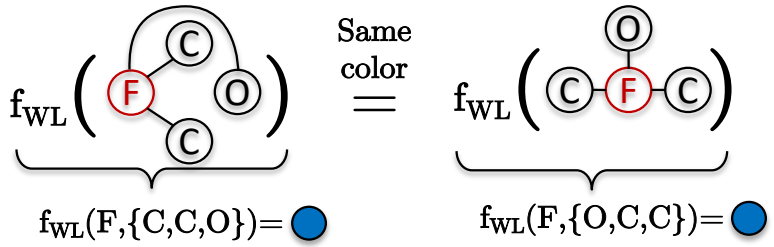
$$f_{\mathrm{WL}}\left(c_i^{\ell}, \{c_j^{\ell}\}_{j \in \mathcal{N}_i}\right) = c_i^{\ell+1}$$

iteration

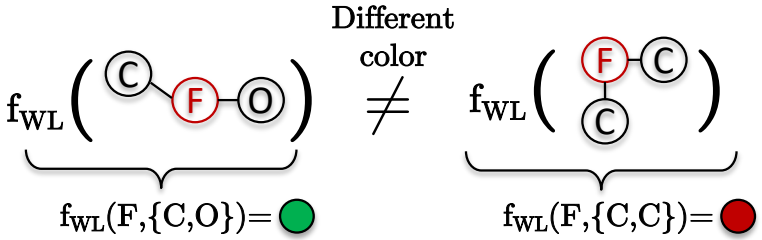
Multiset function

Multiset (set of unordered and repetitive elements)

- Properties of f_{WL} :
  - Unique color for the same pair of (node, its neighborhood).



  - Different colors for distinct pairs of (node, its neighborhood).



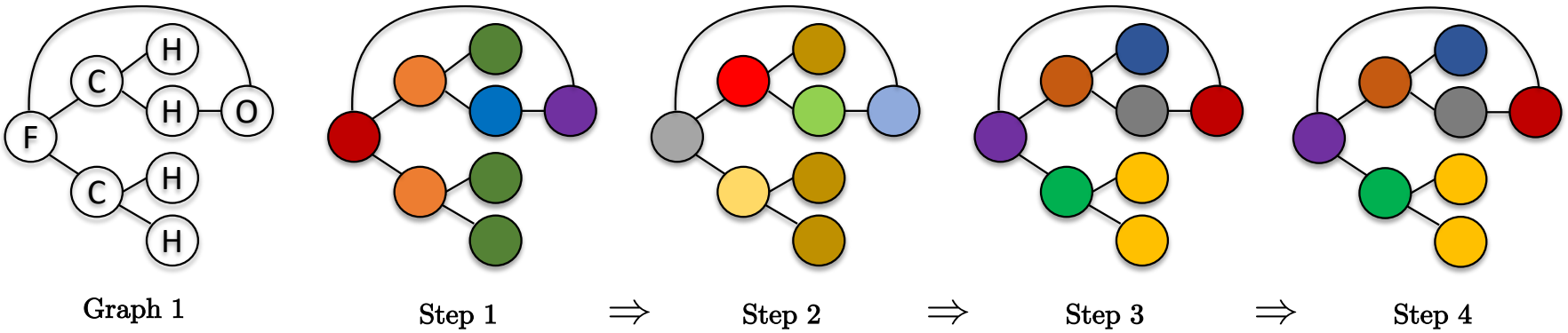
Simply said, f_{WL} must be injective (mapping different inputs to different outputs).

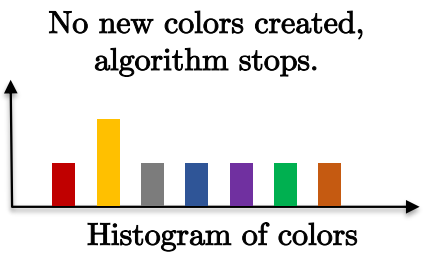
Injectivity is a discriminative property.

[1] Weisfeiler, Lehman, A reduction of a graph to a canonical form and an algebra arising during this reduction, 1968

Weisfeiler-Lehman test

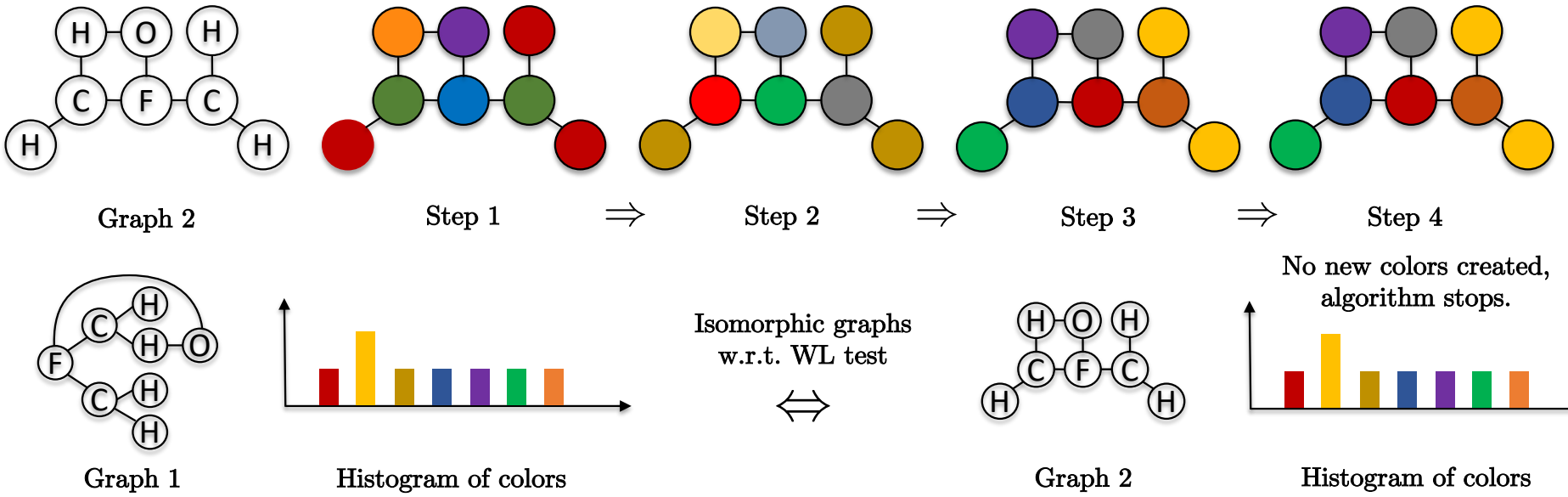
- WL algorithm iteratively applies the coloring function f_{WL} until no new colors are created.
- This produces a canonical representation of a graph as a histogram of colors.





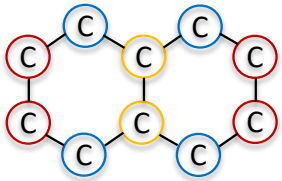
Weisfeiler-Lehman test

- The WL test is a necessary but not sufficient condition for isomorphism.
  - If the two graphs have different color histograms then the two graphs are guaranteed to be non-isomorphic.
  - If the two graphs produce the same histogram of colors, the graphs are possibly isomorphic (but not guaranteed).

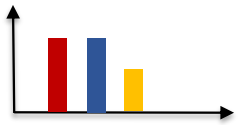


Weisfeiler-Lehman test

- Example of two (simple) non-isomorphic graphs that fail to be distinguished with the WL test.



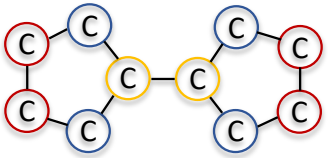
Graph 1
(Decalin)



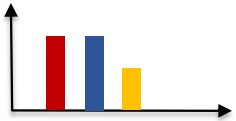
Histogram of colors

Isomorphic graphs
w.r.t. WL test

$\Leftrightarrow$



Graph 2
(Bicyclopentyl)



Histogram of colors

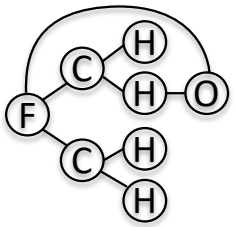
- The original WL test is technically known as the 1-WL test.

Outline

- Weisfeiler-Lehman isomorphism test
- Graph isomorphism networks
- Equivariant graph neural networks
- Expressivity power and generalization
- Equivariant set neural networks
- Conclusion

WL-based graph neural networks

- Can we turn the WL test into a neural network to classify non-isomorphic graphs ?
- Yes, but is it necessary to have a learning process to implement the WL test ? Yes and no.
  - Yes because designing a constraint-free analytical injective function f_{WL} is hard (function f_{WL} as Hash function only provides an approximation).
  - No because neural networks are also limited to provide an approximation of function f_{WL}. However, their approximation can be better than Hash function because of the feature learning mechanism.
- Observe that distinguishing non-isomorphic graphs is not necessarily useful in practice. What is useful is the "coloring" process that can provide expressive representation of data-structured graphs and can be used for downstream tasks like graph classification or regression.



$\Rightarrow$

WL-based GNNs

$\Rightarrow$

Applications :
Classify non-isomorphic graphs
Graph classification/regression

Graph isomorphism networks (GINs)[1]

- The WL test algorithm iterates the "coloring" formula :

$$f_{\mathrm{WL}}\left(c_i^{\ell}, \{c_j^{\ell}\}_{j\in\mathcal{N}_i}\right) = c_i^{\ell+1}$$

  where function f_{WL} provides :

  - A unique color for the same pair $(c_i, \{c_j\}_{\mathcal{N}i})$.
  - Different colors for distinct pairs $(c_i, \{c_j\}_{\mathcal{N}i})$.

  This implies that f_{WL} must be injective :

$$c_i \neq c_{i'} \text{ or } \{c_j^{\ell}\}_{j\in\mathcal{N}_i} \neq \{c_j^{\ell}\}_{j\in\mathcal{N}_{i'}} \;\Rightarrow\; f_{\mathrm{WL}}\left(c_i^{\ell}, \{c_j^{\ell}\}_{j\in\mathcal{N}_i}\right) \neq f_{\mathrm{WL}}\left(c_{i'}^{\ell}, \{c_j^{\ell}\}_{j\in\mathcal{N}_{i'}}\right)$$

- Which aggregator functions are injective?

[1] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019

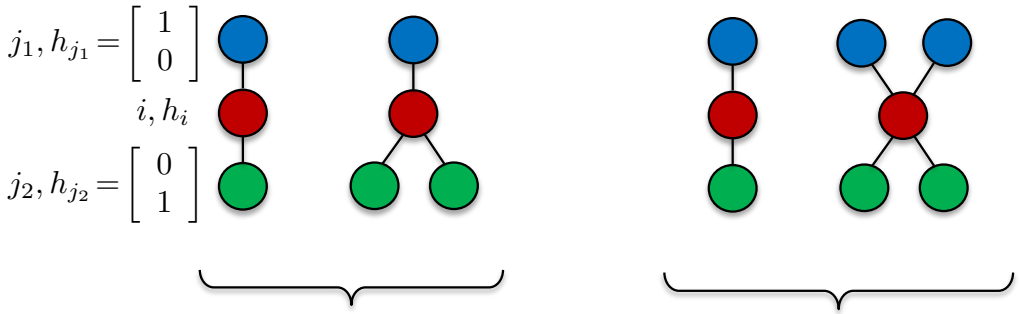
Graph isomorphism networks

- (Simple) aggregator functions f_{WL} :

$$h_i = \max_{j \in \mathcal{N}_i} h_j \qquad \text{Not injective}$$

$$h_i = \text{mean}_{j \in \mathcal{N}_i} h_j \qquad \text{Not injective}$$

$$h_i = \text{sum}_{j \in \mathcal{N}_i} h_j \qquad \text{Injective}$$



Injectivity is a discriminative property at the node level and by extension at the graph level too.

Graph isomorphism networks

- Can we design a GNN function f_{GNN} that is as expressive as the WL test to distinguish non-isomorphic graphs? Yes, this GNN has an aggregation function of the form :

$$f_{\text{GNN}}\left(h_i^\ell, \{h_j^\ell\}_{j\in\mathcal{N}_i}\right) = (1+\varepsilon)g(h_i) + \sum_{j\in\mathcal{N}_i} g(h_j)$$

  where we must have
  - Function g injective.
  - Sum aggregator (mean/max are not injective).
  - Constant ε must be irrational (a number that cannot be written as a ratio of two integers, π = 3.1415..)
- It is difficult to design an analytical injective function g (upper bound on the neighborhood size for countable colors, one-hot encoding does not provide meaningful distance between features).
  - An MLP can approximate g as the universal approximation theorem guarantees the existence of such function (although SGD is not guaranteed to find it):

$$h_i^{\ell+1} = f_{\text{GIN}}\left(h_i^\ell, \{h_j^\ell\}_{j\in\mathcal{N}_i}\right) = \text{MLP}^\ell\left((1+\varepsilon)h_i^\ell + \sum_{j\in\mathcal{N}_i} h_j^\ell\right)$$

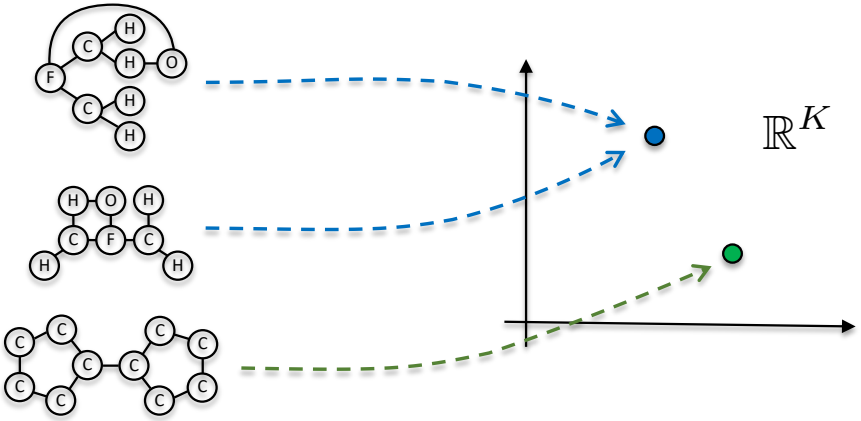
Scalar ε is learned (as computer cannot represent irrational numbers)

Graph isomorphism networks

- Graph readout aggregation function must also be injective :

$$h_{\mathcal{G}} = \mathrm{MLP}\Big(\sum_{i \in V} h_i^L\Big) \in \mathbb{R}^K$$

Sum function



- Summary :
  - Pioneer work[1] with[2] on the theoretical question of expressivity/representation/ discriminative power of GNNs.
  - GINs[1] are designed to be as expressive as the original WL test[3], meaning these networks can learn different graph representations in R^{K} for graphs that can be distinguished by the WL test (coloring and injectivity are the key properties).

[1] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019
[2] Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe, Weisfeiler and leman go neural: Higher-order graph networks, 2019
[3] Weisfeiler, Lehman, A reduction of a graph to a canonical form and an algebra arising during this reduction, 1968

Principal Neighbourhood Aggregation (PNA)[1]

- Graph Isomorphism Networks[2] introduced GNNs are as expressive as the WL test to distinguish non-isomorphic graphs and have an aggregation function of the form :

$$f_{\mathrm{NN}}\left(h_i^\ell, \{h_j^\ell\}_{j\in\mathcal{N}_i}\right) = (1+\varepsilon)g(h_i^\ell) + \sum_{j\in\mathcal{N}_i} g(h_j^\ell)$$

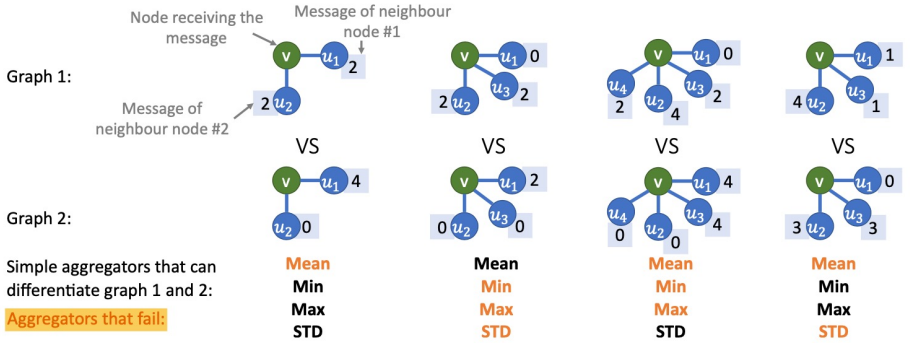
  where function g is injective and sum aggregator is used (mean/max are not injective).

- Principal Neighbourhood Aggregation : Generalization of the injective sum aggregator function to multiple aggregators.
  - Aggregators are mean, max, min and standard deviation.
  - Overall injective function (that guarantees GNN expressivity w.r.t WL test).
  - Increase the expressivity power of standard MP-GNNs while preserving O(N) complexity.

[1] Corso, Cavalleri, Beaini, Lio, and Velickovic, Principal neighbourhood aggregation for graph nets, 2020
[2] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019

Principal Neighbourhood Aggregation

- Illustration of aggregators are mean, max, min and standard deviation :



Reproduced from[1]

- PNA equation :

$$X_i^{\ell+1} = \text{Concat}\left(X_i^\ell, \tilde{X}_i^\ell \; W_1^\ell\right) W_2^\ell \in \mathbb{R}^{d_{\ell+1}}$$

$$\tilde{X}_i^\ell = \oplus_{j \in \mathcal{N}_i}(X_i^\ell, X_j^\ell) \in \mathbb{R}^{12 \times d_\ell}$$

$$= \begin{bmatrix} \text{Mean}_{j \in \mathcal{N}_i}(X_j) \\ \text{Std Dev}_{j \in \mathcal{N}_i}(X_j) \\ \max_{j \in \mathcal{N}_i}(X_j) \\ \min_{j \in \mathcal{N}_i}(X_j) \end{bmatrix} \otimes \begin{bmatrix} \text{Id} \\ S(d,1) \\ S(d,-1) \end{bmatrix}, \quad S(d,\alpha) = \left(\frac{\log(d+1)}{\delta}\right)^\alpha$$

Scalers that amplifies or decreases amplitude of incoming message.

[1] Corso, Cavalleri, Beaini, Lio, and Velickovic, Principal neighbourhood aggregation for graph nets, 2020

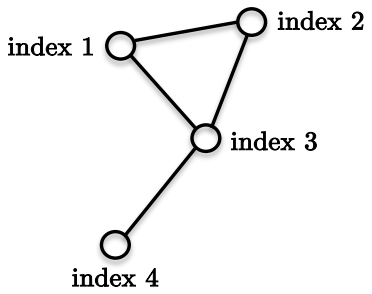
Outline

- Weisfeiler-Lehman isomorphism test
- Graph isomorphism networks
- **Equivariant graph neural networks**
- Expressivity power and generalization
- Equivariant set neural networks
- Conclusion

Graph representations

- Graphs are rank-2 tensors. They can be represented as
  - List of edges (short list for sparse graphs) :

$$\mathrm{E} = [\ \{1,2\},\ \{1,3\},\ \{2,3\},\ \{3,4\}\ ]$$



$$\mathcal{G} = (V, E)$$

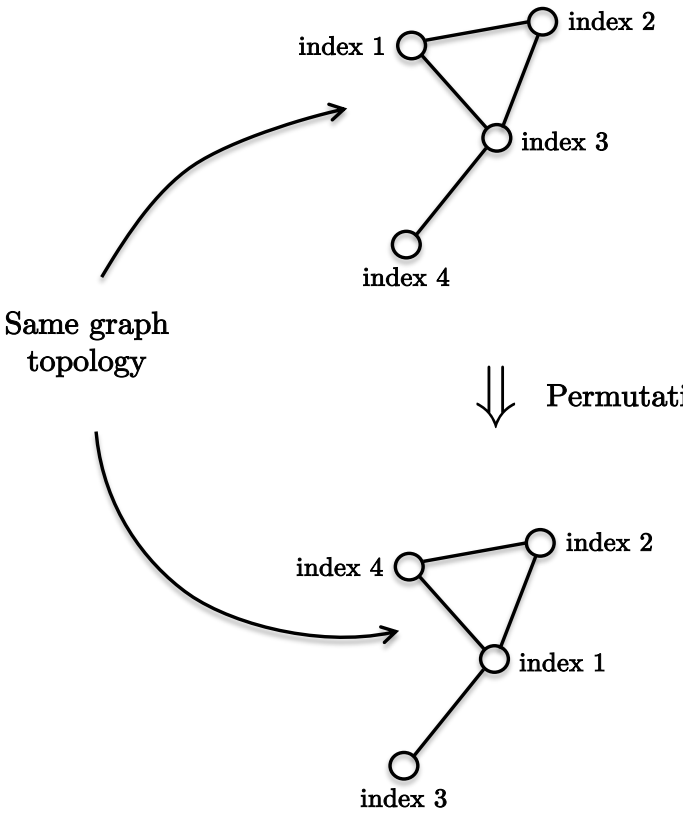
  - Matrix (dense tensor with n being the #nodes) :

$$A = \begin{array}{c} V \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \in \mathbb{R}^{n \times n}$$

- Other tensors :
  - Vectors (ordered and unordered sets) are rank-1 tensors : $\begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} \in \mathbb{R}^n$
  - General tensors (s.a. 3D-MRIs or hyper-graphs) of rank-m :  $\in \mathbb{R}^{n^m}$

Graph symmetry

- The most basic graph symmetry is the invariance of the topology w.r.t. index permutation :



$$V\begin{bmatrix}1&2&3&4\end{bmatrix}\\ \begin{matrix}1\\2\\3\\4\end{matrix}\begin{bmatrix}0&1&1&0\\1&0&1&0\\1&1&0&1\\0&0&1&0\end{bmatrix} = A$$

Different adjacency matrices

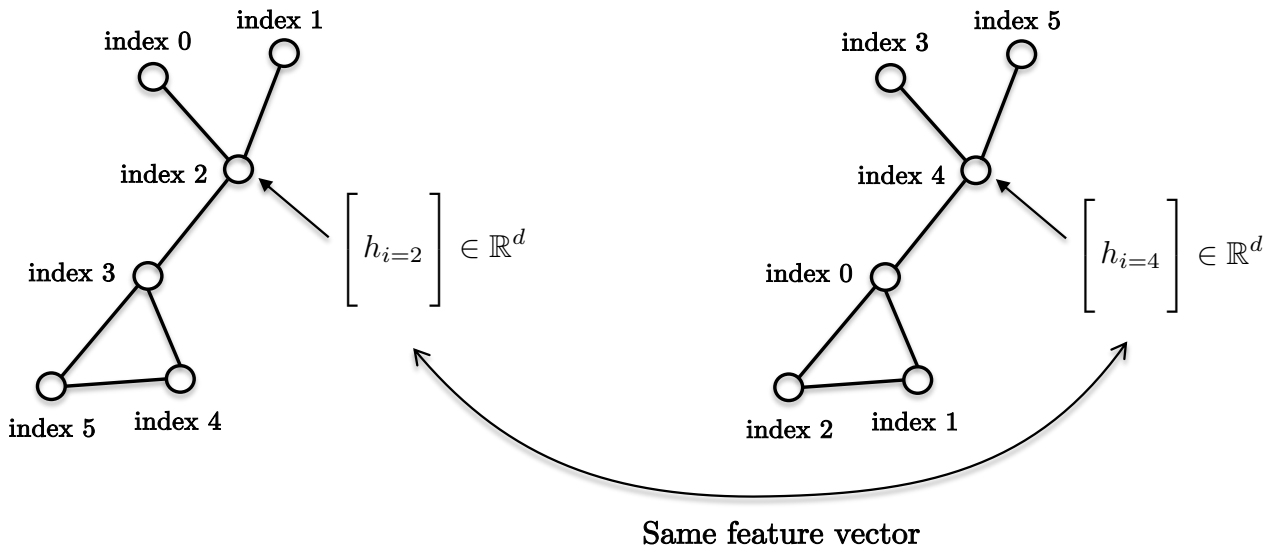
Permutation P

$$\begin{bmatrix}0&0&1&0\\0&1&0&0\\0&0&0&1\\1&0&0&0\end{bmatrix}\begin{bmatrix}0&1&1&0\\1&0&1&0\\1&1&0&1\\0&0&1&0\end{bmatrix}\begin{bmatrix}0&0&0&1\\0&1&0&0\\1&0&0&0\\0&0&1&0\end{bmatrix} = P^T AP = \\ V\begin{bmatrix}1&2&3&4\end{bmatrix}\\ \begin{matrix}1\\2\\3\\4\end{matrix}\begin{bmatrix}0&1&1&1\\1&0&0&1\\1&0&0&0\\1&1&0&0\end{bmatrix}$$

A^{new}

GNNs are permutation invariant

- A built-in property of GNNs is the invariance to index parametrization, i.e. node representation is independent from the choice of node ordering :



- Note that all GNNs s.a. GCNs[1], GAT[2] are permutation invariant.

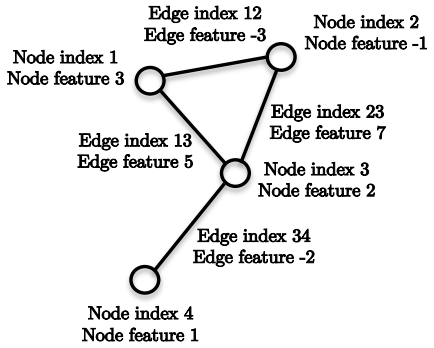
[1] Kipf, Welling, Semi-supervised classification with graph convolutional networks, 2017
[2] Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, Graph attention networks, 2017

Equivariant functions

- What are the functions that preserve index permutation ?
- They exist two types of such functions :
  - Equivariant functions f defined as :

$$\begin{array}{ccc} A \in \mathbb{R}^{n\times n} & \xrightarrow{f} & f(A) \in \mathbb{R}^{n\times n} \\ \downarrow\ P & & \downarrow\ P \\ P^T AP \in \mathbb{R}^{n\times n} & \xrightarrow{f} & f(P^T AP) = P^T f(A) P \in \mathbb{R}^{n\times n} \end{array}$$

  - Equivariant functions model hidden layers. For example, consider A_{ij} to be the feature of edge ij and A_{ii} to be the feature of node i :



$$A = \begin{bmatrix} 3 & -3 & 5 & 0 \\ -3 & -1 & 7 & 0 \\ 5 & 7 & 2 & -2 \\ 0 & 0 & -2 & 1 \end{bmatrix} \quad \text{and} \quad f(A) = A11^T$$

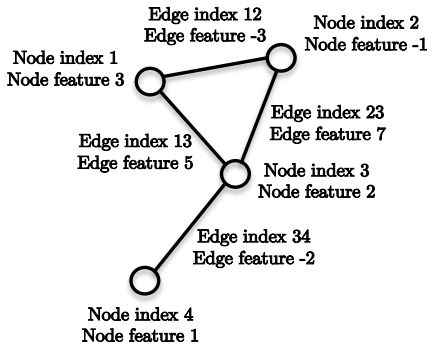
Function f satisfies : $f(P^T AP) = P^T f(A) P$

Invariant functions

- Invariant functions g defined as :

$$\begin{array}{ccc} A \in \mathbb{R}^{n\times n} & \xrightarrow{g} & g(A) \in \mathbb{R}^K, K \geq 1 \\ \downarrow P & & \downarrow P \\ P^T AP \in \mathbb{R}^{n\times n} & \xrightarrow{g} & g(P^T AP) = g(A) \in \mathbb{R}^K, K \geq 1 \end{array}$$

- Invariant functions are readout layers. For example,



$$A = \begin{bmatrix} 3 & -3 & 5 & 0 \\ -3 & -1 & 7 & 0 \\ 5 & 7 & 2 & -2 \\ 0 & 0 & -2 & 1 \end{bmatrix} \quad \text{and} \quad g(A) = 1^T A 1 = \sum_{ij} A_{ij}$$

Function g satisfies : $g(P^T AP) = g(A)$

Symmetry & equivariance

- The most basic image symmetry is translation.
  - Image content remains unchanged w.r.t. translations.
- What is the function that preserves translations ?
  - Convolution (translation equivariant function)
  - Convolutional neural networks[1] are translation equivariant (but not rotation equivariant).
- Generalization[2,3] :
  - Group convolution × symmetry (e.g. rotation+translation, mesh convolution)
- Why symmetry/equivariance matter ?
  - Great inductive biases
    - Less parameters to learn.
    - Faster training, better generalization.

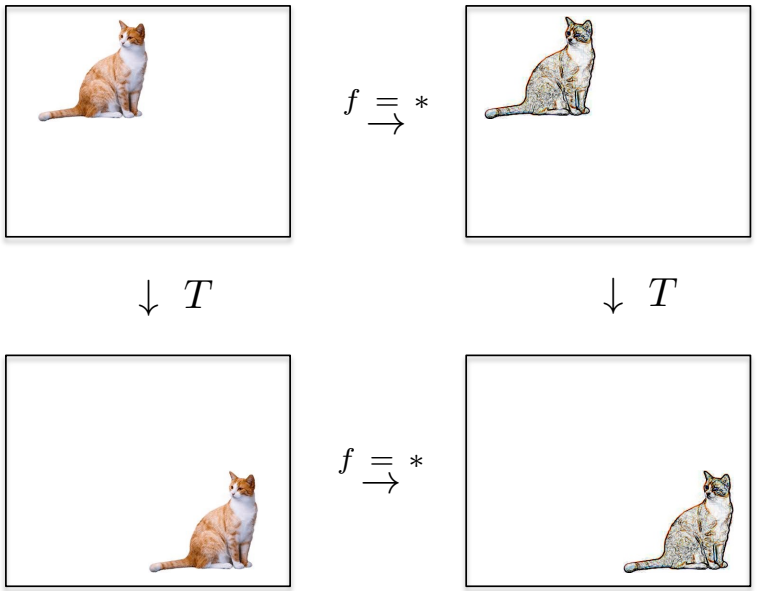


Image symmetry : Translation
Equivariant function : Convolution
(with a line detector kernel)

$$f(x) = x * k, \quad k = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

[1] LeCun, Bottou, Bengio, Haffner, Gradient-based learning applied to document recognition, 1998
[2] Cohen, Welling, Group Equivariant Convolutional Networks, 2016
[3] de Haan, Weiler, Cohen, Welling, Gauge Equivariant Mesh CNNs: Anisotropic convolutions on geometric graphs, 2020

MLP (for vectors)

- What is the simplest neural network that can be defined for (ordered) vectors ?
  - An MLP : Composition of { linear functions + non-linear activations/ReLU } + final linear layer for classification/regression :

$$y = m_W(x) = g_{W^L} \circ \sigma \circ f_{W^{L-1}} \circ \sigma \circ f_{W^{L-2}} \circ ... \circ \sigma \circ f_{W^0} \circ x$$

Output vector

$y \in \mathbb{R}^K, K \geq 1$

Classification/ regression layer

$g_W(x) = Wx + b \in \mathbb{R}^K, K \geq 1$

L linear layers

$f_W(x) = Wx + b \in \mathbb{R}^n$

Input vector

$x \in \mathbb{R}^n$

MLP for graphs[1]

- What is the simplest neural network that can be defined for graph adjacency matrices $A \in \mathrm{R}^{\mathrm{n\ x\ n}}$?
  - Equivariant GNNs (E-GNNs)[1] : Composition of { equivariant linear functions + non-linear activations/ReLU } + invariant linear function + final linear function for classification/regression :

$$y = m_{W^{L+1}} \circ g_{W^L} \circ \sigma \circ f_{W^{L-1}} \circ \sigma \circ f_{W^{L-2}} \circ ...\sigma \circ f_{W^0} \circ A$$

Output vector

$y \in \mathbb{R}^K, K \geq 1$

MLP Classification/ regression layer

$m_W(x) \in \mathbb{R}^K, K \geq 1$

Invariant linear layer

$g_W(P^T AP) = g_W(A) \in \mathbb{R}^K, K \geq 1$

L equivariant linear layers

$f_W(P^T AP) = P^T f_W(A) P \in \mathbb{R}^{n \times n}$

Input graph matrix

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

[1] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019

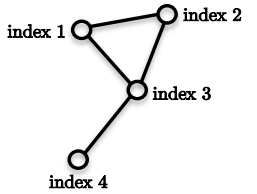
Linear equivariant/invariant spaces

- Given a rank-2 tensor $A \in R^{n \times n}$, can we fully characterize all equivariant/invariant linear functions $f : R^{n \times n} \to R^{n \times n}$ / $g : R^{n \times n} \to R^{K}$?
  - In other words, what is the biggest collection of equivariant/invariant linear layers ?
  - This is to achieve maximal expressivity of linear functions invariant by permutation.
- Theorem[1] : The space of all equivariant linear functions is fully characterizable and of dimension Bell(4)=15, and the space of all invariant linear functions is of dimension Bell(2)=2.
- Note that the number of functions is independent of the number of nodes n.
  - These functions can be used with datasets of graphs with variable sizes.
- General case[1] : Given an order-m tensor $A \in R^{n^m}$, the space of all equivariant linear functions $f : R^{n^m} \to R^{n^{m'}}$ is of dimension Bell(m+m') and the space of all invariant linear functions $g : R^{n^m} \to R^{K}$ is of dimension K.Bell(m).

[1] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019

Linear equivariant functions

- The 15 equivariant linear functions $f_k : R^{n \times n} \rightarrow R^{n \times n}$ for a given tensor $A \in R^{n \times n}$ are defined as :
  - The identity and transpose operations: $f_1(A) = A,\ f_2(A) = A^T$
  - The diag operation: $f_3(A) = \text{diag}(\text{diag}(A))$
  - Sum of rows replicated on rows/ columns/ diagonal:
    $$f_4(A) = A11^T,\ f_5(A) = 1(A1)^T,\ f_6(A) = \text{diag}(A1),\ 1 \in \mathbb{R}^n$$
  - Sum of columns replicated on rows/ columns/ diagonal:
    $$f_7(A) = A^T11^T,\ f_8(A) = 1(A^T1)^T,\ f_9(A) = \text{diag}(A^T1)$$
  - Sum of all elements replicated on all matrix/ diagonal:
    $$f_{10}(A) = (1^TA1) \cdot 11^T,\ f_{11}(A) = (1^TA1) \cdot \text{diag}(1)$$
  - Sum of diagonal elements replicated on all matrix/diagonal:
    $$f_{12}(A) = (1^T\text{diag}(A)) \cdot 11^T,\ f_{13}(A) = (1^T\text{diag}(A)) \cdot \text{diag}(1)$$
  - Replicate diagonal elements on rows/columns:
    $$f_{14}(A) = \text{diag}(A)1^T,\ f_{15}(A) = 1\text{diag}(A)^T$$
  - Note that some f_k (A) tensors are dense, although A is sparse.



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

$$f_1(A) = \begin{bmatrix} 0&1&1&0 \\ 1&0&1&0 \\ 1&1&0&1 \\ 0&0&1&0 \end{bmatrix} \quad f_2(A) = \begin{bmatrix} 0&1&1&0 \\ 1&0&1&0 \\ 1&1&0&1 \\ 0&0&1&0 \end{bmatrix}$$

$$f_3(A) = \begin{bmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix} \quad f_4(A) = \begin{bmatrix} 2&2&2&2 \\ 2&2&2&2 \\ 3&3&3&3 \\ 1&1&1&1 \end{bmatrix}$$

$$f_5(A) = \begin{bmatrix} 2&2&3&1 \\ 2&2&3&1 \\ 2&2&3&1 \\ 2&2&3&1 \end{bmatrix} \quad f_6(A) = \begin{bmatrix} 2&0&0&0 \\ 0&2&0&0 \\ 0&0&3&0 \\ 0&0&0&1 \end{bmatrix}$$

$$f_7(A) = \begin{bmatrix} 2&2&2&2 \\ 2&2&2&2 \\ 3&3&3&3 \\ 1&1&1&1 \end{bmatrix} \quad f_8(A) = \begin{bmatrix} 2&2&3&1 \\ 2&2&3&1 \\ 2&2&3&1 \\ 2&2&3&1 \end{bmatrix}$$

etc

Equivariant layer parametrization

- From 1-dim input to 1-dim output :

15 equivariant functions

2 bias functions

$$A \in \mathbb{R}^{n\times n} \;\rightarrow\; f_W(A) = \sum_{k=1}^{15+2} W_k \, f_k(A) \;\in \mathbb{R}^{n\times n}$$

17 (learnable) parameters

- From d-dim input to 1-dim output :

$$A \in \mathbb{R}^{n\times n\times d} \;\rightarrow\; f_W(A) = \sum_{k=1}^{15+2} \sum_{q=1}^{d} W_{k,q} \, f_k(A_{\cdot,\cdot,q}) \;\in \mathbb{R}^{n\times n}$$

17 x d (learnable) parameters

Sliced tensor A along the third dimension.

- From d-dim input to d'-dim output :

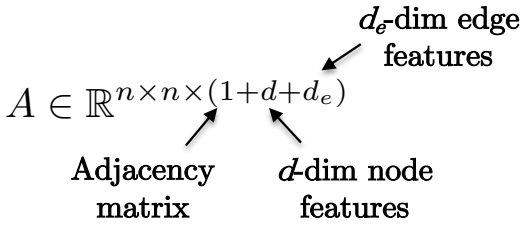
$$A \in \mathbb{R}^{n\times n\times d} \;\rightarrow\; \big(f_W(A)\big)_{\cdot,\cdot,q'} = \sum_{k=1}^{15+2} \sum_{q=1}^{d} W_{k,q,q'} \, f_k(A_{\cdot,\cdot,q}) \;\in \mathbb{R}^{n\times n\times d'}$$

17 x d x d' (learnable) parameters

- Memory/speed complexity is $O(n^2)$.

Equivariant GNNs[1]

- Start with :



- Forward pass of E-GNNs :

$$y = \underbrace{m_{W^{L+1}}}_{\text{MLP layer}} \circ \underbrace{g_{W^L}}_{\text{Invariant layer}} \circ \sigma \circ \underbrace{f_{W^{L-1}} \circ \sigma \circ f_{W^{L-2}} \circ \ldots \sigma \circ \circ f_{W^0}}_{L \text{ Equivariant layers}} \circ A$$

with $f_{W^\ell} : \mathbb{R}^{n \times n \times d^\ell} \to \mathbb{R}^{n \times n \times d^{\ell+1}}$

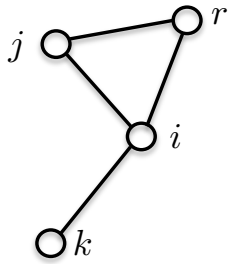
$g_{W^L} : \mathbb{R}^{n \times n \times d^L} \to \mathbb{R}^K$

- #Parameters/layer is : $17 \times d^\ell \times d^{\ell+1}$
- Memory/speed complexity is O(n^2).
- E-GNNs are the most expressive linear networks for graphs w.r.t. the original 1-WL test.
  - However, the 1-WL test fails to distinguish simple non-isomorphic graphs.

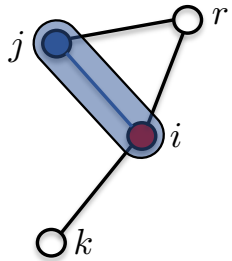
[1] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019
[2] Gilmer, Schoenholz, Riley, Vinyals, Dahl, Neural message passing for quantum chemistry, 2017

Boosting the expressivity power

- Can we design more expressive GNNs, i.e. GNNs that are better than the original WL test ?
  - k-WL tests : Original test uses 2-tuple of nodes to produce colors. To produce more colors, and improve expressivity power of WL tests, higher-order interactions between nodes with k-tuple of nodes with k ≥ 3 can be used.

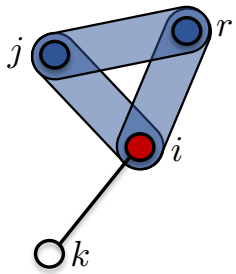


$\mathcal{G} = (V, E)$



Edges $= \{i, j\}, \{i, r\}, \{i, k\}$

2-tuple of nodes can distinguish non-isomorphic graphs with the 1-WL/2-WL tests.



Hyper-edges $= \{i, j, r\}, \{i, k, r\}, \{i, j, k\}$

3-tuple of nodes can distinguish non-isomorphic graphs with the 3-WL test.

Higher expressivity power

- What is the expressivity power of equivariant GNNs in terms of WL tests ?
  - Let us define a k-order Equivariant GNNs :

$$y = m_{W^{L+1}} \circ g_{W^L} \circ \sigma \circ f_{W^{L-1}} \circ \sigma \circ f_{W^{L-2}} \circ ...\sigma \circ f_{W^0} \circ A$$
$$\text{with } k = \max_{\ell \in [0, L-1]} k_\ell$$
$$\text{and } f_{W^\ell} : \mathbb{R}^{n^{k_\ell} \times d_\ell} \to \mathbb{R}^{n^{k_{\ell+1}} \times d_{\ell+1}}$$

- Theorem[1] : There exists a k-order E-GNN that can distinguish non-isomorphic graphs with the k-WL test.
  - Existence result (although SGD is not guaranteed to find such network).
  - It is not a practical result as k-order E-GNNs requite $O(n^k)$ memory/speed complexities.
    - Note that 1-WL/2-WL GNNs (s.a. GINs) have the same discriminative power[2]. It implies that we must have at least k=3 to be more powerful than GINs. But then, it requires $O(n^3)$.

[1] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019
[2] Cai, Furer, Immerman, An optimal lower bound on the number of variables for graph identification, 1992

3-WL GNNs[1]

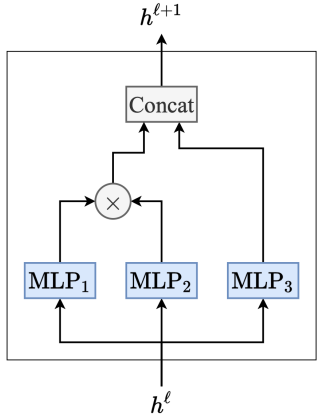
- How to design GNNs that are provable 3-WL but do not require $O(n^3)$ memory/speed complexities ?
- 3-WL GNNs[1] : To achieve higher interactions between nodes, it was proposed to multiply matrices (that generates non-local interactions).
  - Theorem : There exists a 3-WL GNN as expressive as the 3-WL test.
  - Memory is quadratic $O(n^2)$ but matrix-matrix multiplication implies $O(n^3)$ speed.
    - It improves memory complexity compared to 3-order E-GNNs[2].
    - Matrix-matrix multiplication densifies sparse matrix.

[1] Maron, Ben-Hamu, Serviansky, Lipman, Provably powerful graph networks, 2019
[2] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019

3-WL GNNs

- **3-WL GNN update layer :**

$$h^{\ell+1} = \text{Concat}\big(m_{W_1^\ell}(h^\ell) \cdot m_{W_2^\ell}(h^\ell),\ m_{W_3^\ell}(h^\ell)\big),$$
$$\text{where } h^\ell, h^{\ell+1} \in \mathbb{R}^{n\times n\times d}, W_a, W_b \in \mathbb{R}^{d\times d},$$



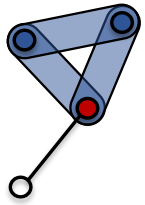
where m_W are 2-layer MLPs applied along the feature dimension. The matrix-matrix multiplication is carried out along the first and second dimensions as :

$$\big(M_{W_1}(h)\ .\ M_{W_2}(h)\big)_{i,j,k} = \sum_{p=1}^{n} \big(M_{W_1}(h)\big)_{i,p,k} \cdot \big(M_{W_2}(h)\big)_{p,j,k}$$

with complexity $O(n^3)$.

Sparse k-WL-GNNs[1]

- Original k-WL GNNs are provable as expressive as the k-WL test, but they are not practical because of $O(n^k)$ memory/speed requirements.
  - Dense rank-k tensors represent all possible k-tuple of nodes (existing or not).
  - These GNNs do not leverage graph sparsity, which is known to be a good inductive bias for generalization. Also, dense tensors increase over-fitting.
- Sparse k-WL-GNNs[1] solely consider the k-tuple of nodes present in the given graphs.
  - Theorem[1] : Sparse k-WL-GNNs are strictly more powerful than dense k-WL-GNNs.
  - Theorem[1,2] : Sparse k-WL-GNNs exhibit a smaller generalization error than dense k-WL-GNNs.
  - In practice, the memory and speed complexities are improved, and benefit from the graph sparsity. However, the numerical time still remains large as all k-tuple of nodes present in the graph must be considered at each layer.

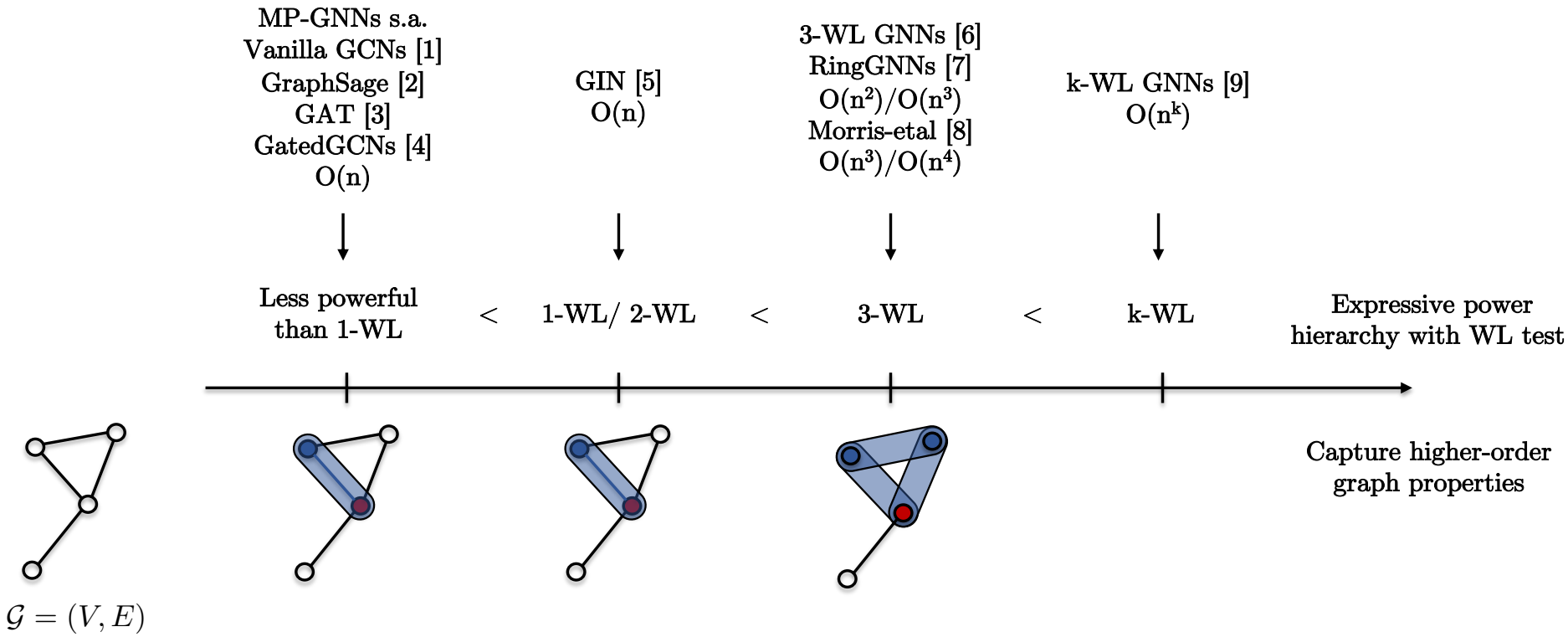


[1] Morris, Rattan, Mutzel, Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings, 2020
[2] Garg, Jegelka, Jaakkola, Generalization and representational limits of graph neural networks, 2020

Outline

- Weisfeiler-Lehman isomorphism test
- Graph isomorphism networks
- Equivariant graph neural networks
- **Expressivity power and generalization**
- Equivariant set neural networks
- Conclusion

Expressivity power w.r.t WL test



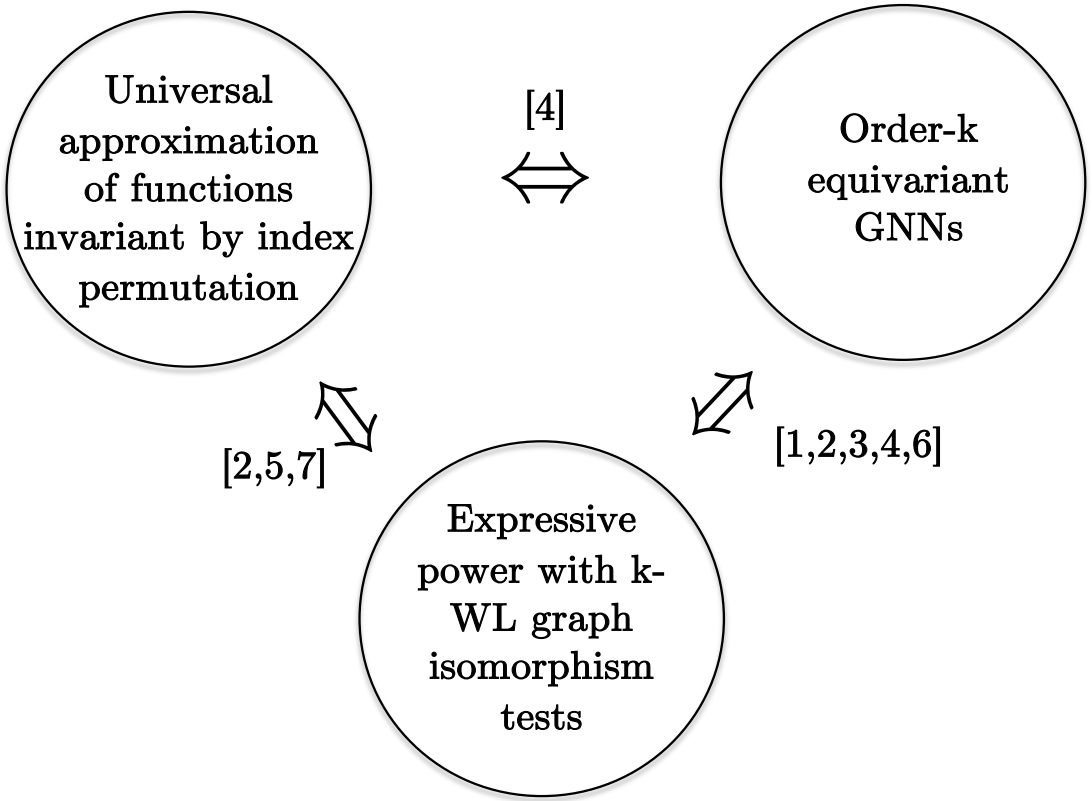
[1] Kipf, Welling, Semi-supervised classification with graph convolutional networks, 2017
[2] Hamilton, Ying, Leskovec, Inductive representation learning on large graphs, 2017
[3] Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, Graph attention networks, 2017
[4] Bresson, Laurent, Residual gated graph convnets, 2017
[5] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019
[6] Maron, Ben-Hamu, Serviansky, Lipman, Provably powerful graph networks, 2019
[7] Chen, Villar, Chen, Bruna, On the equivalence between graph isomorphism testing and function approximation with gnns, 2019
[8] Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe, Weisfeiler and leman go neural: Higher-order graph neural networks, 2019
[9] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019

Universal approximator

- What functions can be approximated by MLP ?
  - Universal approximation theorem[1,2] : Any continuous function can be arbitrarily approximated with a MLP, with the necessary condition that the number of hidden neurons goes to infinity (or very large).
- What functions can be approximated by Equivariant GNNs ?
  - Theorem[3] : Any continuous function invariant by permutation can be arbitrarily approximated by E-GNNs, with the necessary condition the network has a tensor of order m=poly(n)=n(n-1)/2, where n is the number of nodes.
  - High-order tensors are not practical as memory/speed complexities grow as $O(n^m)$.
- Result in [3] is for the worst case and was improved in [4] to be independent of n (but no upper bound on the order is known).

[1] Cybenko, Approximation by superpositions of a sigmoidal function, 1989
[2] Hornik, Approximation capabilities of multilayer feedforward networks, 1991
[3] Maron Fetaya, Segol, Lipman, On the universality of invariant networks, 2019
[4] Keriven, Peyré, Universal invariant and equivariant graph neural networks, 2019

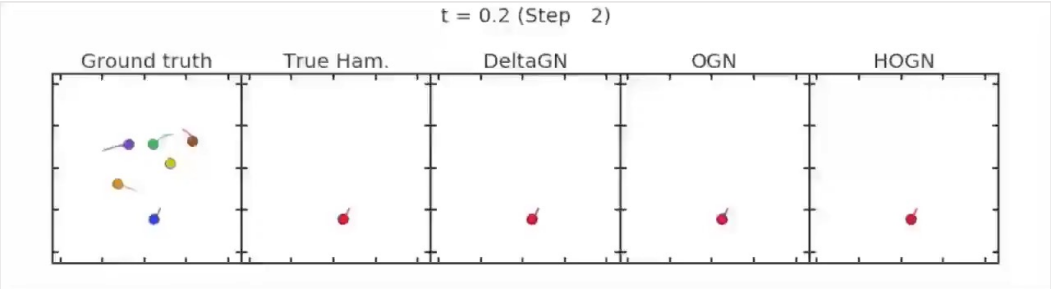
Relationships

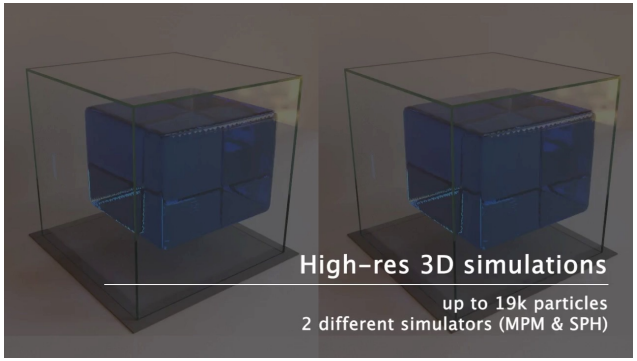


[1] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019
[2] Maron Fetaya, Segol, Lipman, On the universality of invariant networks, 2019
[3] Keriven, Peyré, Universal invariant and equivariant graph neural networks, 2019
[4] Maron, Ben-Hamu, Serviansky, Lipman, Provably powerful graph networks, 2019
[5] Chen, Villar, Chen, Bruna, On the equivalence between graph isomorphism testing and function approximation with gnns, 2019
[6] Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe, Weisfeiler and leman go neural: Higher-order graph neural networks, 2019
[7] Loukas, What graph neural networks cannot learn: depth vs width, 2019

Universal approximator and generalization

- Universal approximator for functions defined on graphs and expressiveness/representation power to distinguish non-isomorphic graphs are necessary conditions to design good GNNs.
- However, these properties do not guarantee the best generalization performance. It is important to make the distinction between representation power and generalization!
  - MLP are universal approximators for general functions but they do not generalize well.
  - Equivariant GNNs are MLP for permutation-invariant functions but they do not generalize as well as MP-GNNs s.a. GAT/GatedGCNs[1].
- MP-GNNs which benefit from tailored inductive biases can lead to strong generalization performance.
  - For example, combining MP-GNNs with physical constraints like Hamiltonian mechanics[2,3].

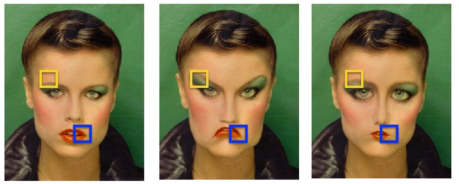




[1] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020
[2] Sanchez-Gonzalez, Bapst, Cranmer, Battaglia, Hamiltonian Graph Networks with ODE Integrators, 2019
[3] Sanchez-Gonzalez, Godwin, Pfaff, Ying, Leskovec, Battaglia, Learning to simulate complex physics with graph networks, 2020

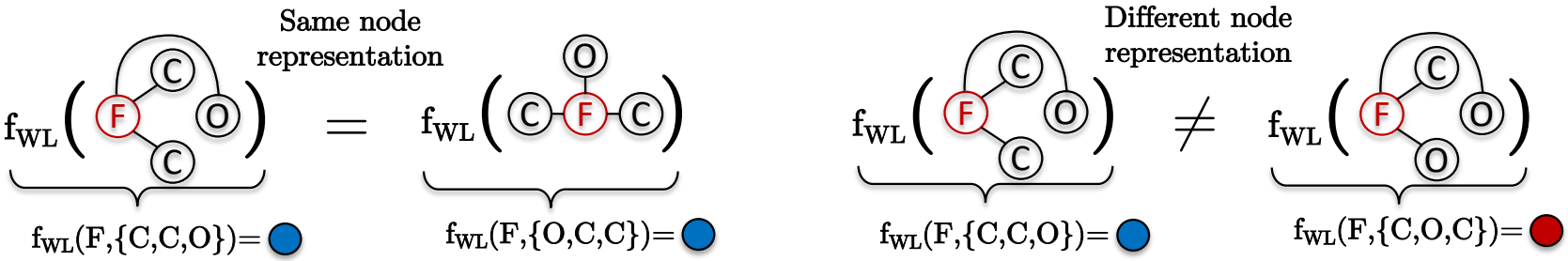
Question on isomorphism expressivity

- CNNs are robust to local deformation invariance (diffeomorphism on grids). This property is critical for generalization.



Reproduced from Bruna's NYU lectures

- Are WL-GNNs robust to local graph deformations?
  - The WL test focused on finding distinct node representation for different neighborhoods.
  - Two nodes must have two different representations if their neighborhoods are distinct just by a single neighbor.
  - But a representation robust to perturbations (i.e. unchanged or very close) is desirable for generalization.

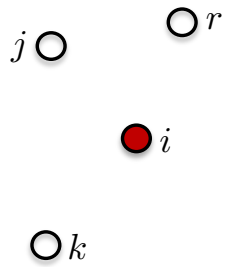
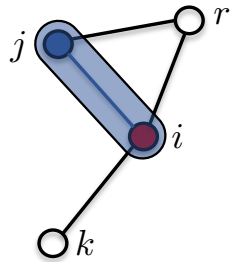
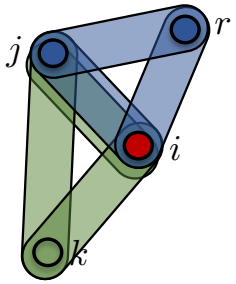


Outline

- Weisfeiler-Lehman isomorphism test
- Graph isomorphism networks
- Equivariant graph neural networks
- Expressivity power and generalization
- **Equivariant set neural networks**
- Conclusion

Beyond graphs

- Tensors invariant by permutation :

Sets	Graphs	Tensors
First-order tensors (Unordered) vectors 1-tuple of nodes	Second-order tensors Graphs 2-tuple of nodes	Order-m tensors Hyper-graphs m-tuple of nodes
		
No edge	Edges $= \{i,j\},\{i,r\},\{i,k\}$	Hyper-edges $=\{i,j,r\}$, $\{i,k,r\},\{i,j,k\}$

- Equivariant GNNs[1] can be run on any tensor of order-m, with memory/speed complexities $O(n^{\max(m,m')})$.

$$f_W : \mathbb{R}^{n^m} \to \mathbb{R}^{n^{m'}}$$
$$g_W : \mathbb{R}^{n^m} \to \mathbb{R}^K, K \geq 1$$

[1] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019

Equivariant/invariant linear layer for sets

- For rank-1 tensors/vectors, the equivariant linear transformation $f_W : R^n$ to R^n is fully characterized by bell(2)=2 equivariant functions, which are I and 11^T-I. The parametrized equivariant linear transformation is[1,2,3] :

$$f_W(x) = \left[w_1 \mathrm{I} + w_2 \left(11^T - \mathrm{I}\right)\right] x \in \mathbb{R}^n$$

$$f_W(x)_i = w_1 x_i + w_2 \sum_{j \neq i} x_j$$

$$f_W\left(P \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = f_W\left(\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = f_W\left(\begin{bmatrix} x_3 \\ x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} w_1 x_3 + w_2(x_1 + x_2) \\ w_1 x_1 + w_2(x_2 + x_3) \\ w_1 x_2 + w_2(x_1 + x_3) \end{bmatrix} = P f_W\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right)$$

- The invariant linear transformation $g_W : R^n$ to R^K is fully characterized by K.bell(1)=K equivariant function, which is $1_{K \times n}$. The parametrized invariant linear transformation is

$$g_W(x) = w_1 1_{K \times n} x \in \mathbb{R}^K, K \geq 1$$

- Linear equivariant functions for sets have limited performance (only 2 parameters to learn per layer), and thus cannot be performant for NLP tasks (sentences are rank-1 tensors).

[1] Zaheer, Kottur, Ravanbakhsh, Poczos, Salakhutdinov, Smola, Deep sets, 2017
[2] Qi, Su, Mo, Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017
[3] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019

Equivariant/invariant linear layer for sets

- How to improve the performance of linear equivariant functions for sets ?
  - Consider non-linear equivariant functions.
    - Transformers[1] use (non-linear) attention functions to produce node representations that adapt to the context.
    - Breakthrough in NLP

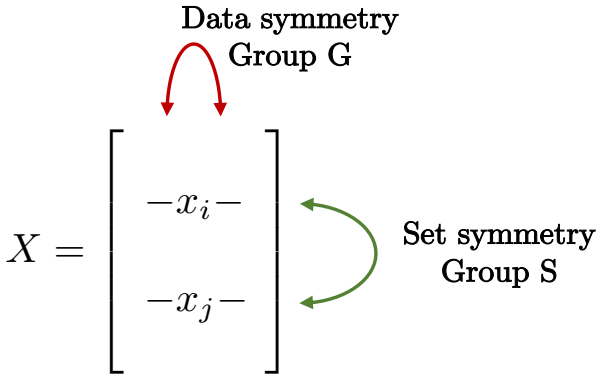
$$h_i^{\ell+1} = f_W(A) = \sum_{j \in V} \frac{\exp(W_1 h_j^\ell)}{\sum_{k \in V} \exp(W_2 h_k^\ell)} W_3 h_j^\ell$$

  - Consider additional symmetry/invariance[2,3].

[1] Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin, Attention is all you need, 2017
[2] Zaheer, Kottur, Ravanbakhsh, Poczos, Salakhutdinov, Smola, Deep sets, 2017
[3] Maron, Litany, Chechik, Fetaya, On Learning Sets of Symmetric Elements, 2020

On Learning Sets of Symmetric Elements[1]

- Neural networks for sets of symmetric data.
  - Set symmetry : order invariance
  - Data symmetry : translation invariance for example
- A set of data can be represented as a matrix $X \in R^{n \times d}$



- DeepSets[2]/PointNets[3] : Neural network for arbitrary sets of data points (order invariance only).
  - These set networks are universal for permutation invariant functions. DeepSets is also universal for equivariant functions[4].

$$f^S_W(X)_i = f^L_{W_1}(x_i) + f^L_{W_2}(\sum_{j \neq i} x_j)$$

$$f^L_{W_1}(z) = w_1 z, \;\; f^L_{W_2}(z) = w_2 z$$

- ConvNets[5] : Neural network for symmetric data (translation invariance only)

$$f^G_W(x_i) = W * x_i$$

[1] Maron, Litany, Chechik, Fetaya, On Learning Sets of Symmetric Elements, 2020
[2] Zaheer, Kottur, Ravanbakhsh, Poczos, Salakhutdinov, Smola, Deep sets, 2017
[3] Qi, Su, Mo, Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017
[4] Segol, Lipman, On universal equivariant set networks, 2019
[5] Maron, Litany, Chechik, Fetaya, On Learning Sets of Symmetric Elements, 2020

On Learning Sets of Symmetric Elements

- Deep Sets for Symmetric elements (DSS)[1] : Combining symmetries from DeepSets and ConvNets into a single framework.

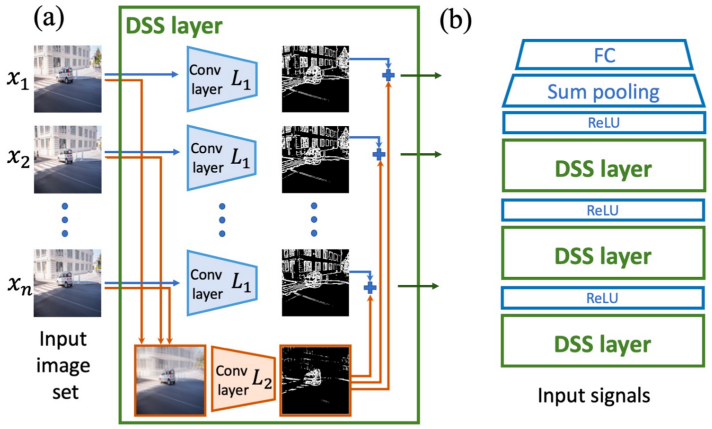
- Any linear equivariant layer for the symmetry group S × G is of the form :

$$f_W^{S\times G}(X)_i = f_{W_1}^G(x_i) + f_{W_2}^G(\sum_{j\neq i} x_j),$$

$$f_W^G(x) = W * x$$

- Expressivity power : If G-equivariant networks are universal, then S × G-equivariant networks are also universal, that is it can represent any function with the S × G symmetry.
  - Note that DSS expressivity is lower than DeepSets (symmetry group is larger) but DSS are as expressive as they can be for this symmetry group.
  - Experimental generalization performance is better with this symmetry group.

- Applications : set of 1D-signals, 3D cloud points and set of images.



Outline

- Weisfeiler-Lehman isomorphism test
- Graph isomorphism networks
- Equivariant graph neural networks
- Expressivity power and generalization
- Equivariant set neural networks
- **Conclusion**

Conclusion

- WL test is used as expressivity measure of representation power of GNNs, but it is not the only expressivity measure. For example, we can define expressivity as the capacity of identifying/counting the number of elementary sub-structures like cycles and cliques[1].
- WL-GNNs are computationally more expensive[2] than standard MP-GNNs, i.e. $O(n^k)$ vs $O(E)$.
- Expressivity does not necessarily imply generalization[3], like MLP.
- Equivariance/invariance w.r.t. group of transformations can be powerful to generalize if it captures the right data properties, otherwise it will (strongly) overfit.

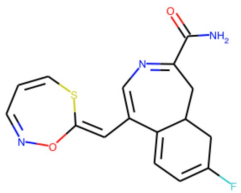


Fig 1. Graph with cycles (rings)

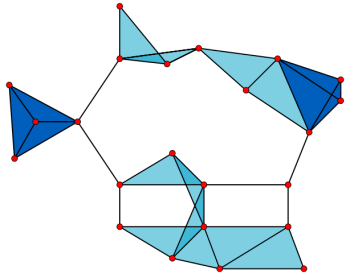


Fig 2. Graph with
19 × 3-vertex cliques (light and dark blue triangles)
and 2 × 4-vertex cliques (dark blue areas)

[1] Chen, Chen, Villar, Bruna, Can graph neural networks count substructures? 2020
[2] Maron, Ben-Hamu, Serviansky, Lipman, Provably powerful graph networks, 2019
[3] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020

Questions?