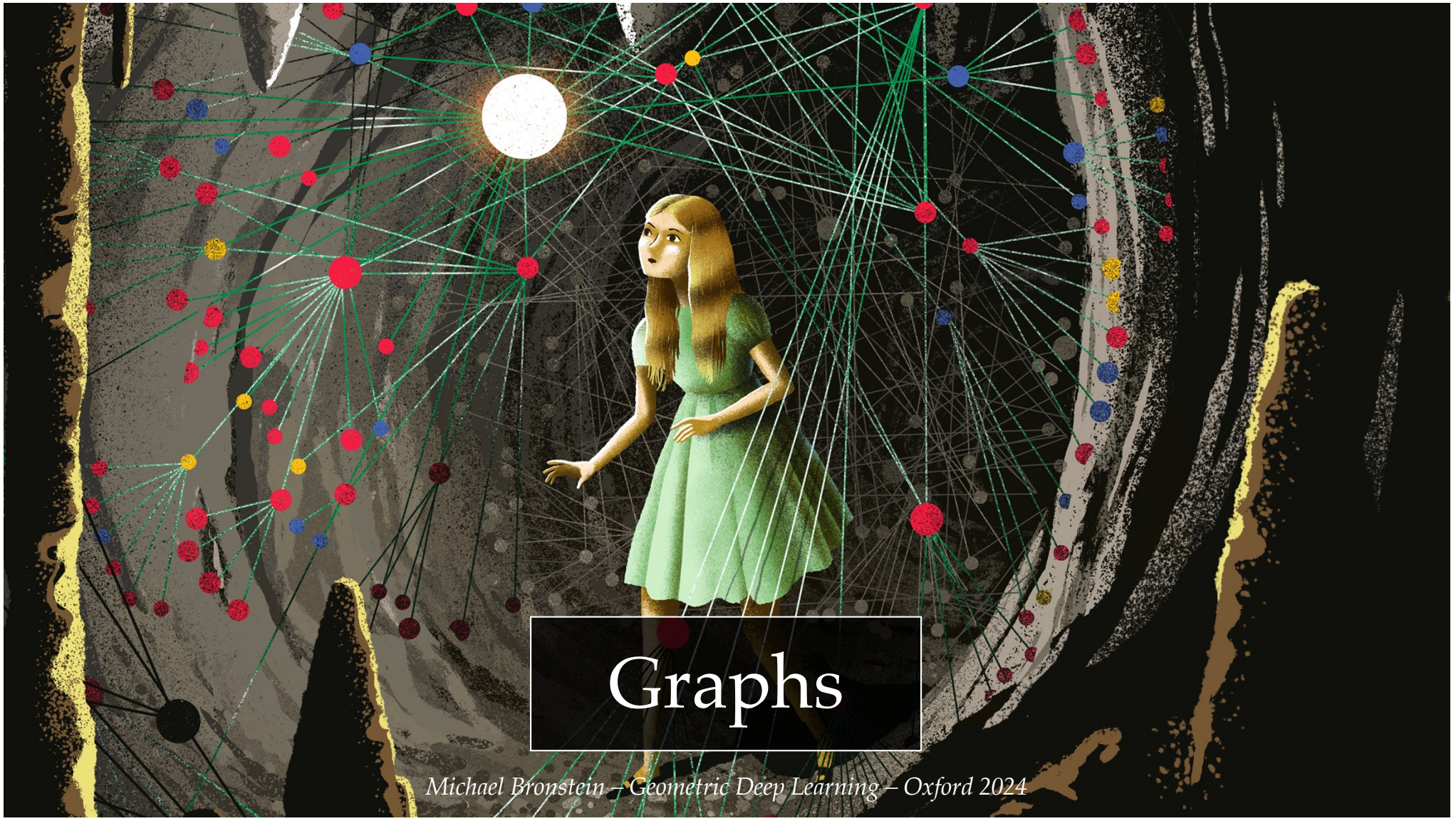
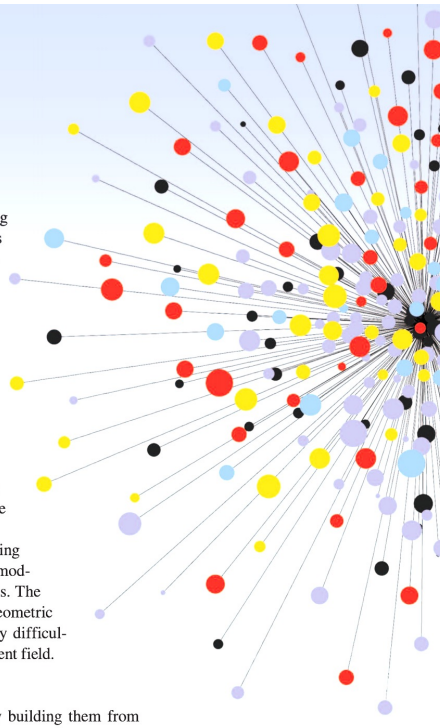

Graphs
Michael Bronstein – Geometric Deep Learning – Oxford 2024

Outline

- We now move to Sets to Graphs
- Sets can be considered as “node-only” Graphs without edges
- Graphs are a very versatile abstraction used ubiquitously in sciences
- For many years, Geometric Deep Learning was nearly synonymous with Graph Representation Learning

Michael M. Bronstein, Joan Bruna, Yann LeCun,
Arthur Szlam, and Pierre Vandergheynst



Many scientific fields study data with an underlying structure that is non-Euclidean. Some examples include social networks in computational social sciences, sensor networks in communications, functional networks in brain imaging, regulatory networks in genetics, and meshed surfaces in computer graphics. In many applications, such geometric data are large and complex (in the case of social networks, on the scale of billions) and are natural targets for machine-learning techniques. In particular, we would like to use deep neural networks, which have recently proven to be powerful tools for a broad range of problems from computer vision, natural-language processing, and audio analysis. However, these tools have been most successful on data with an underlying Euclidean or grid-like structure and in cases where the invariances of these structures are built into networks used to model them.

Geometric deep learning is an umbrella term for emerging techniques attempting to generalize (structured) deep neural models to non-Euclidean domains, such as graphs and manifolds. The purpose of this article is to overview different examples of geometric deep-learning problems and present available solutions, key difficulties, applications, and future research directions in this nascent field.

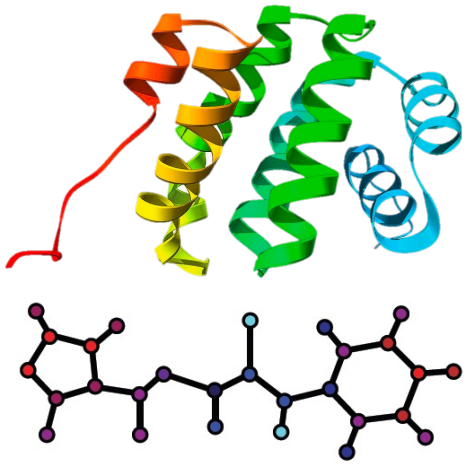
Overview of deep learning

Deep learning refers to learning complicated concepts by building them from simpler ones in a hierarchical or multilayer manner. Artificial neural networks are popular realizations of such deep multilayer hierarchies. In the past few years, the growing computational power of modern graphics processing unit (GPU)-based computers and the availability of large training data sets have allowed successfully training neural networks with many layers and degrees of freedom (DoF) [1]. This has led to qualitative breakthroughs on a wide variety of tasks, from speech recognition [2], [3] and machine translation [4] to image analysis and computer vision [5]–[11] (see [12]

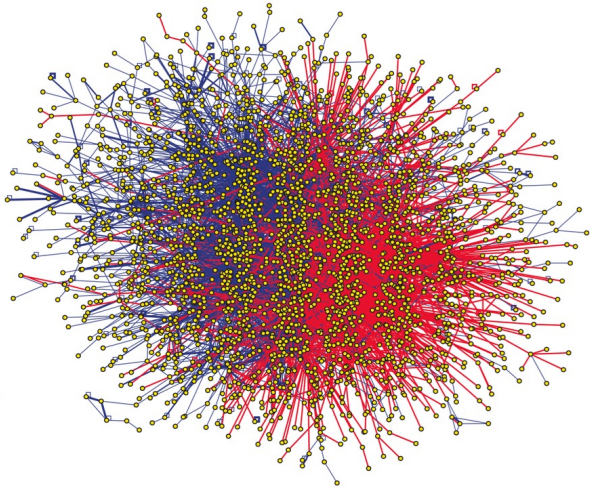
Geometric Deep Learning

Going beyond Euclidean data

Graphs = Systems of Relations and Interactions



Molecules



Interactomes

Social networks

Why Graphs?

- Simple mathematical abstraction for systems of relations or interactions
- Very well developed theory
- Many successful applications from chemistry to social media to even pure math
- In many ways, graphs are the main modality of data we receive from Nature
- Graph representation learning is likely critical on the path to Artificial General Intelligence (AGI).

Why Graphs?

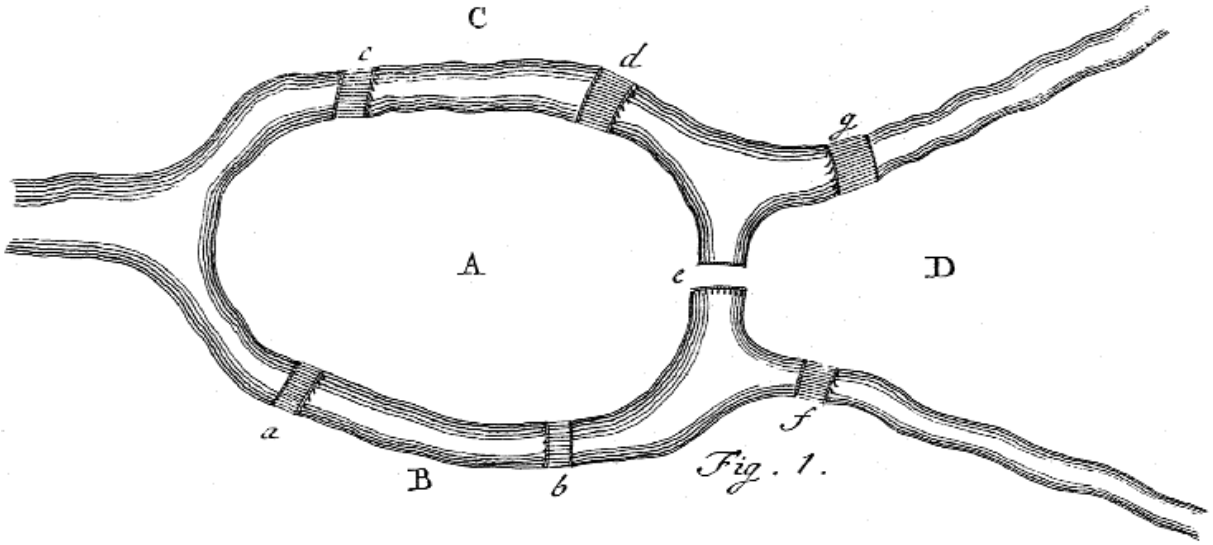
"The image of the world around us, which we carry in our head, is just a *model*. Nobody in his head imagines all the world, government or country. He has only selected *concepts*, and *relationships* between them, and uses those to represent the real system."

—Jay Forrester on mental models (1971)

Jay Wright Forrester

Forrester 1971

Origins of Graph Theory



L. Euler

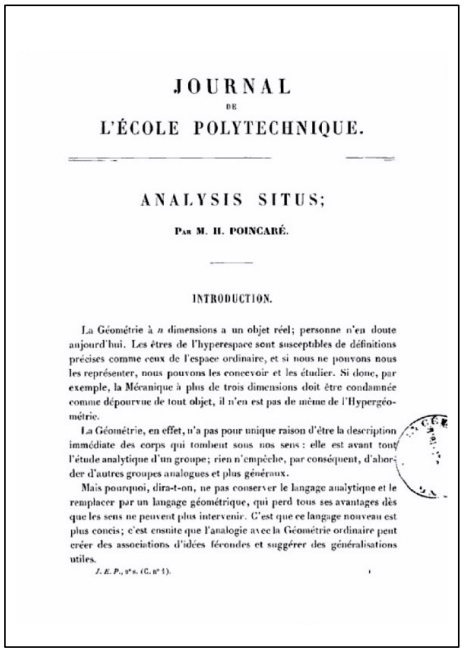
1741

The soluton of the classical problem of the "Bridges of Königsberg" by Euler in 1736 first showed the power of graphs to abstract out the geometry (*"geometria situs"*)

Euler 1741

Origins of Topology

Poincaré's *"analysis situs."*
His famous Conjecture
appeared in a supplement
published in 1904.



JOURNAL
DE
L'ÉCOLE POLYTECHNIQUE.

ANALYSIS SITUS;

Par M. H. POINCARÉ.

INTRODUCTION.

La Géométrie à n dimensions a un objet réel; personne n'en doute aujourd'hui. Les êtres de l'hyperespace sont susceptibles de définitions précises comme ceux de l'espace ordinaire, et si nous ne pouvons nous les représenter, nous pouvons les concevoir et les étudier. Si donc, par exemple, la Mécanique à plus de trois dimensions doit être condamnée comme dépourvue de tout objet, il n'en est pas de même de l'Hypergéométrie.

La Géométrie, en effet, n'a pas pour unique raison d'être la description immédiate des corps qui tombent sous nos sens : elle est avant tout l'étude analytique d'un groupe; rien n'empêche, par conséquent, d'aborder d'autres groupes analogues et plus généraux.

Mais pourquoi, dira-t-on, ne pas conserver le langage analytique et le remplacer par un langage géométrique, qui perd tous ses avantages dès que les sens ne peuvent plus intervenir. C'est que ce langage nouveau est plus concis; c'est ensuite que l'analogie avec la Géométrie ordinaire peut créer des associations d'idées fécondes et suggérer des généralisations utiles.

J. E. P., 2e s. (C. n° 1).

H. Poincaré

1895

L. Euler

1741

Euler 1741; Poincaré 1895

Origin of the name "Graph"

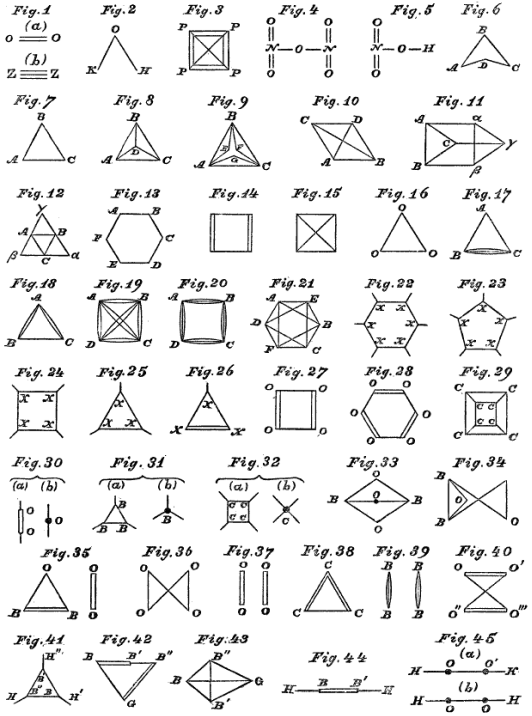
CHEMISTRY AND ALGEBRA

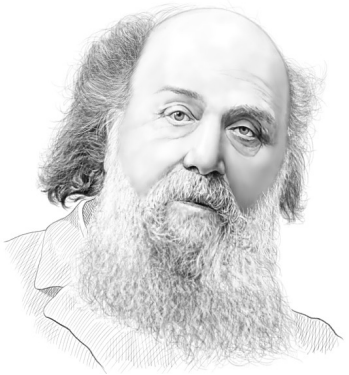
IT may not be wholly without interest to some of the readers of NATURE to be made acquainted with an analogy that has recently forcibly impressed me between branches of human knowledge apparently so dissimilar as modern chemistry and modern algebra.

The weight of an invariant is identical with the number of the bonds in the chemicograph of the analogous chemical substance, and the weight of the leading term (or basic differentiant) of a co-variant is the same as the number of bonds in the chemicograph of the analogous compound radical. Every invariant and covariant thus becomes expressible by a *graph* precisely identical with a Kekuléan diagram or chemicograph.

Baltimore, January 1 J. J. SYLVESTER

The term "graph" appeared first in the chemical context



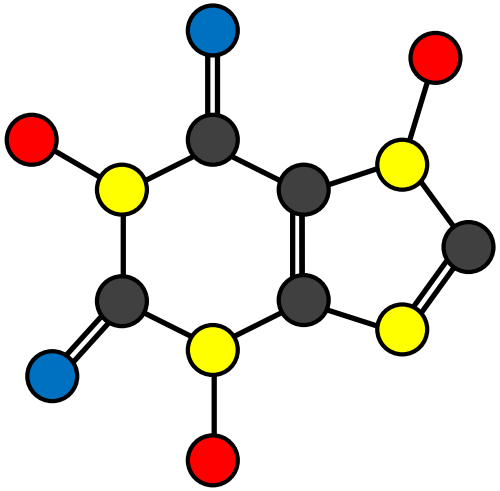


J. Sylvester

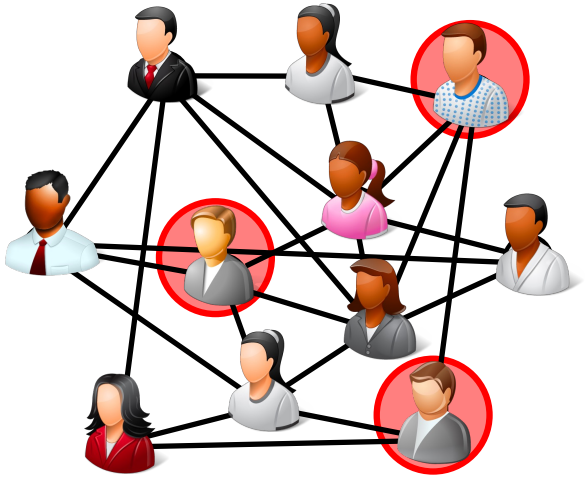
1878

GRAPHS: THE BASICS

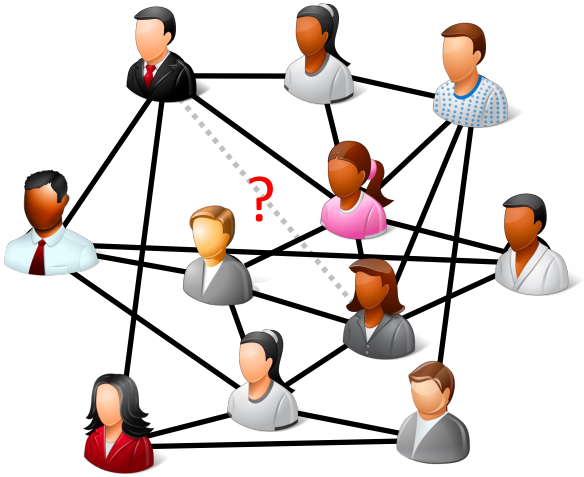
Types of Learning Tasks on Graphs



Graph regression
water solubility?

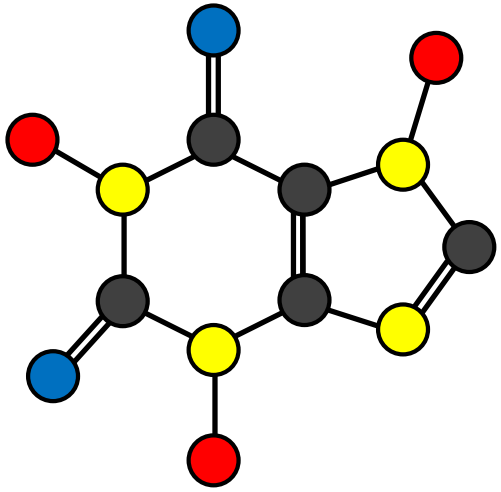


Node classification
who is a spammer

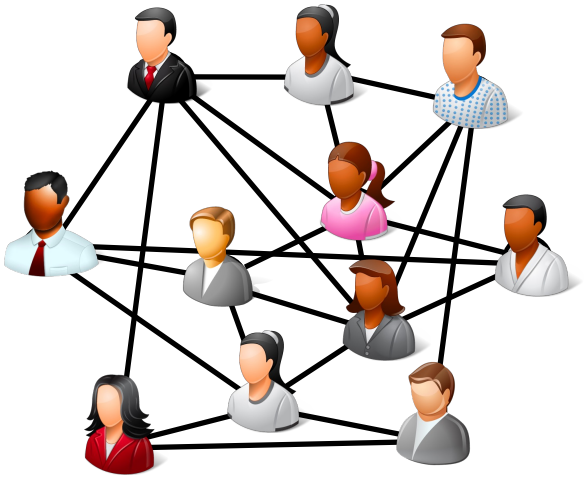


Link prediction
WTF (whom to follow)?

Invariant vs Equivariant tasks

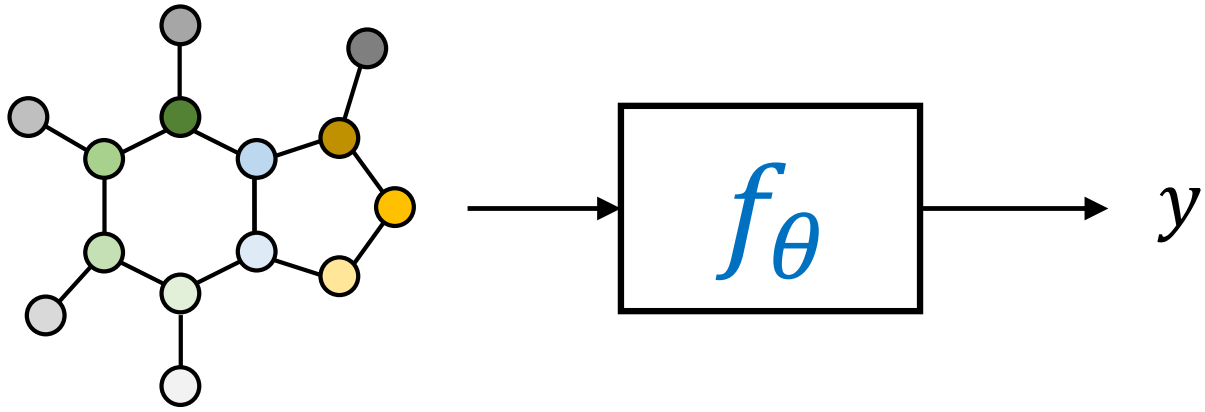


water solubility?



who is a spammer?

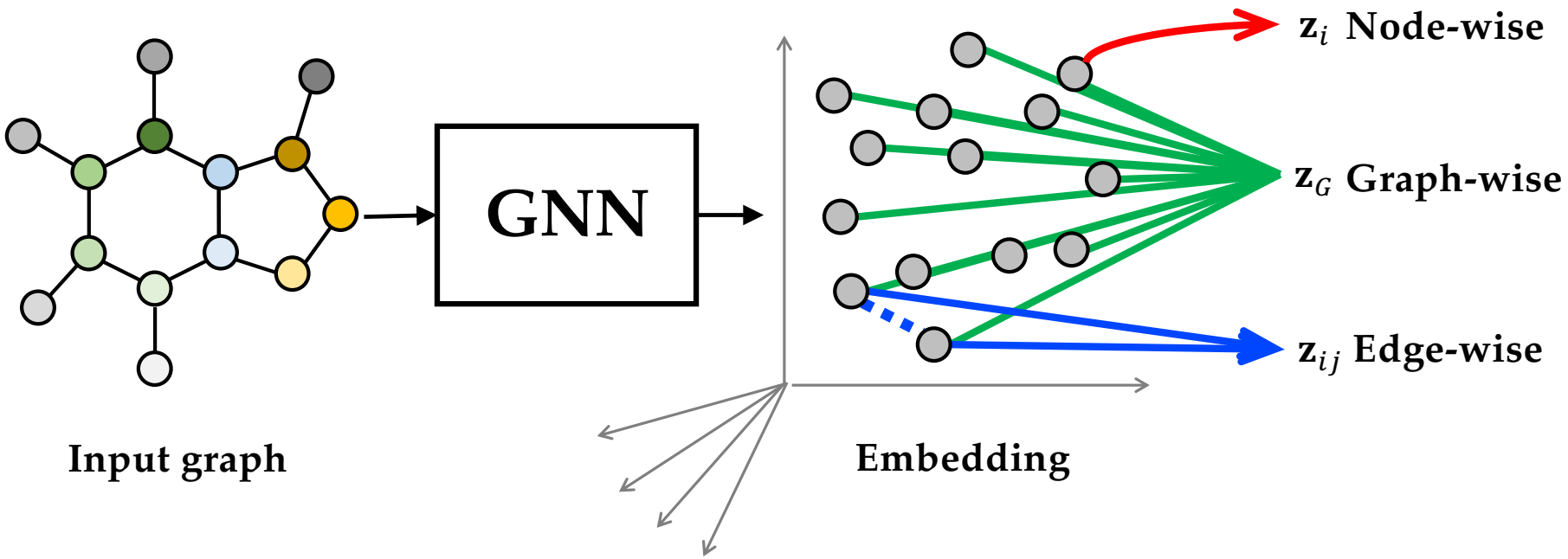
GNNs = Parametric graph functions



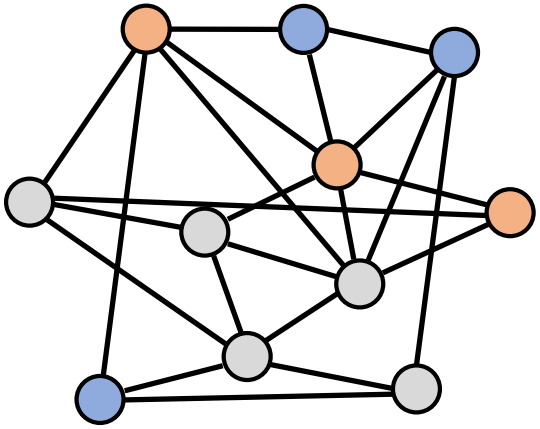
Graph+Features

Note: we use the term "GNN" to refer to general parametric graph functions. The particular class of GNNs we consider are Message Passing Neural Networks (MPNNs). Petar Veličković argues that "it's all message passing"

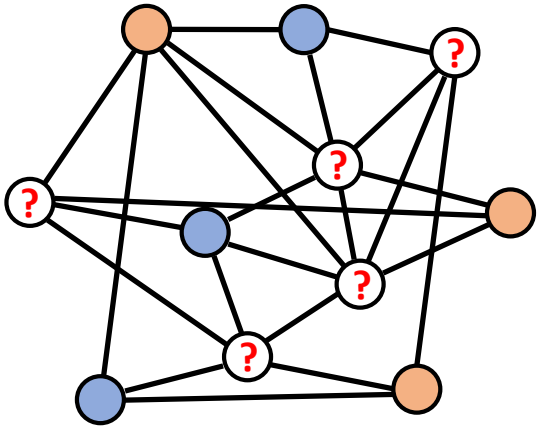
Types of Learning Tasks on Graphs



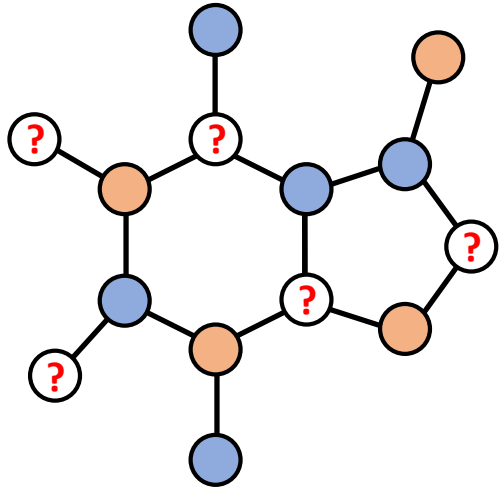
Transductive vs Inductive tasks



Training



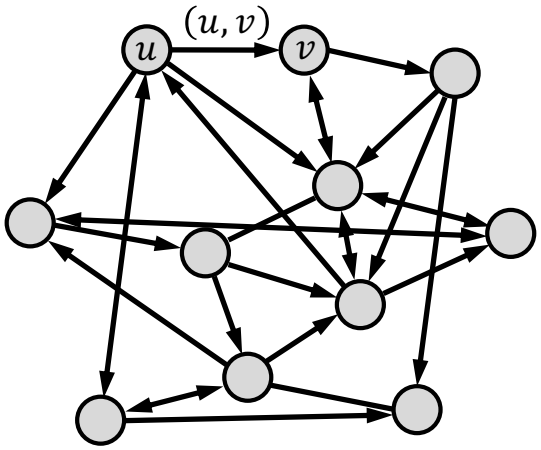
Transductive
same graph,
different nodes



Inductive
different graph

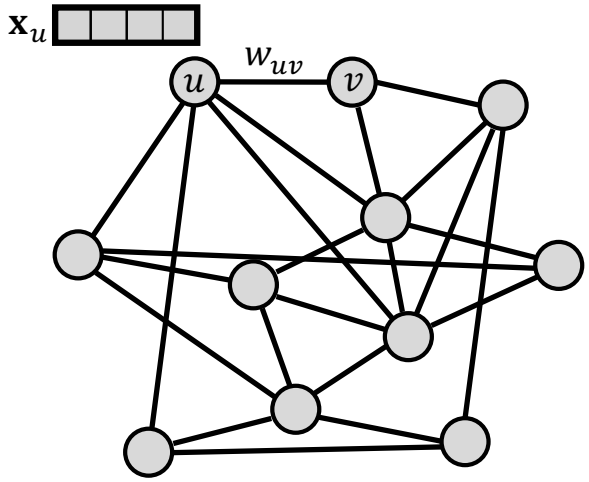
Graphs: The Basics

- A **graph** is a pair $G = (V, E)$
  - V = **vertices** (or **nodes**)
  - $E \subseteq V \times V$ = **edges** (or **links**). If $(u, v) \in E$, we will write $u \sim v$
- Edges in general are **directed**, i.e., we might have $u \sim v$ but not $v \sim u$.

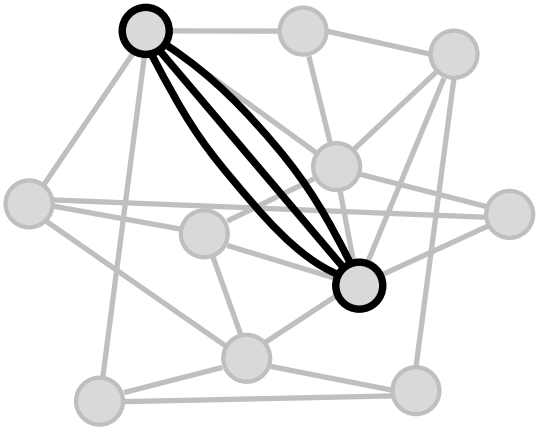


Graphs: The Basics

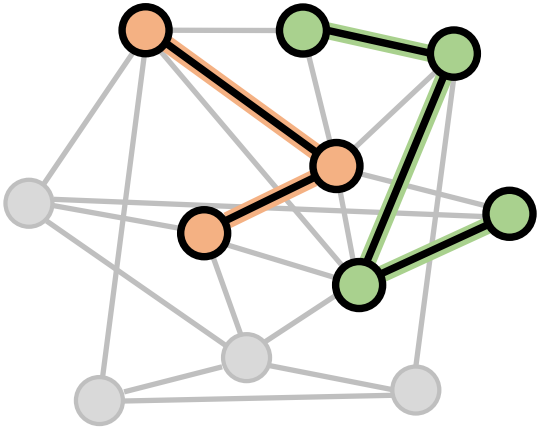
- A **graph** is a pair $G = (V, E)$
  - V = **vertices** (or **nodes**)
  - $E \subseteq V \times V$ = **edges** (or **links**). If $(u, v) \in E$, we will write $u \sim v$
- Edges in general are **directed**, i.e., we might have $u \sim v$ but not $v \sim u$.
- Nodes and edges can have **attributes**. We will typically assume
  - d-dimensional vector **node features**, denoted $\mathbf{x}_u$
  - scalar **edge weights**, denoted w_{uv}



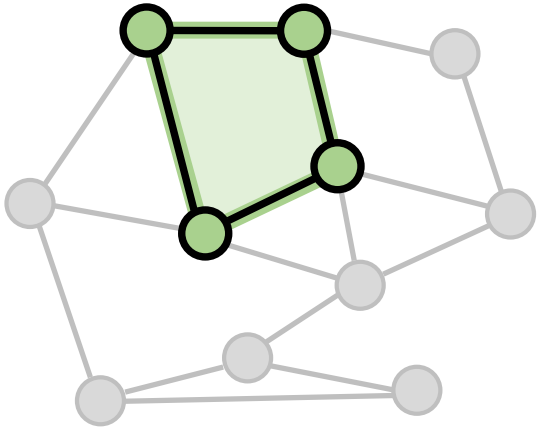
Extensions of Graphs



Multigraph

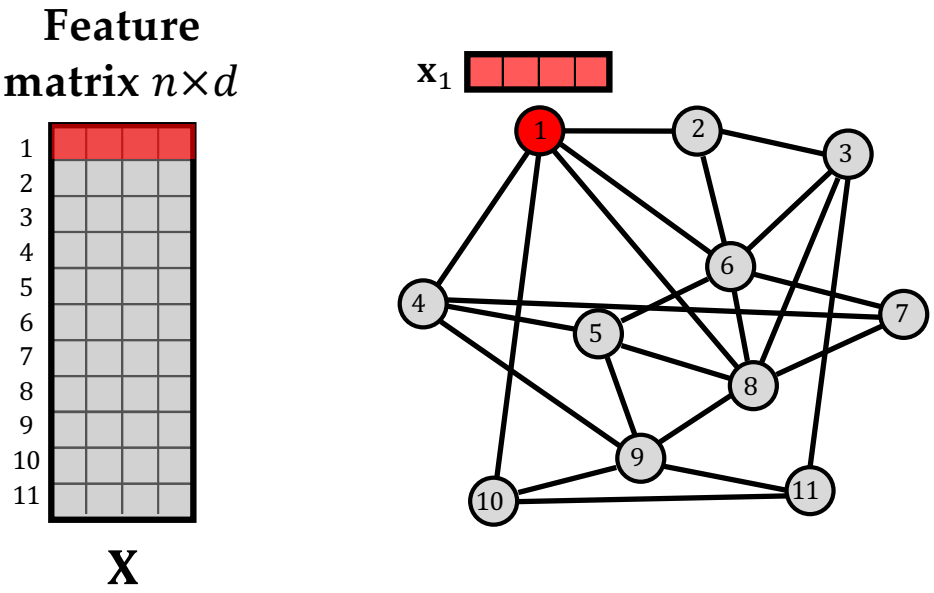


Hypergraph



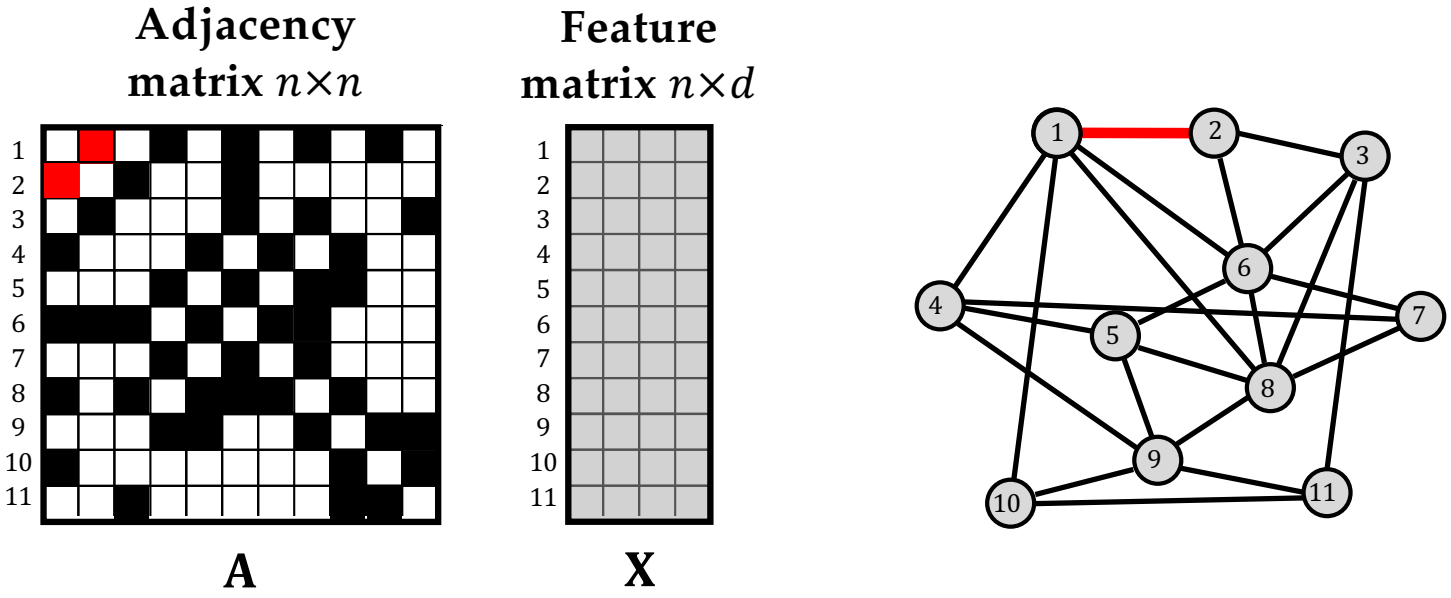
Cellular complex

Representations of Graphs as matrices



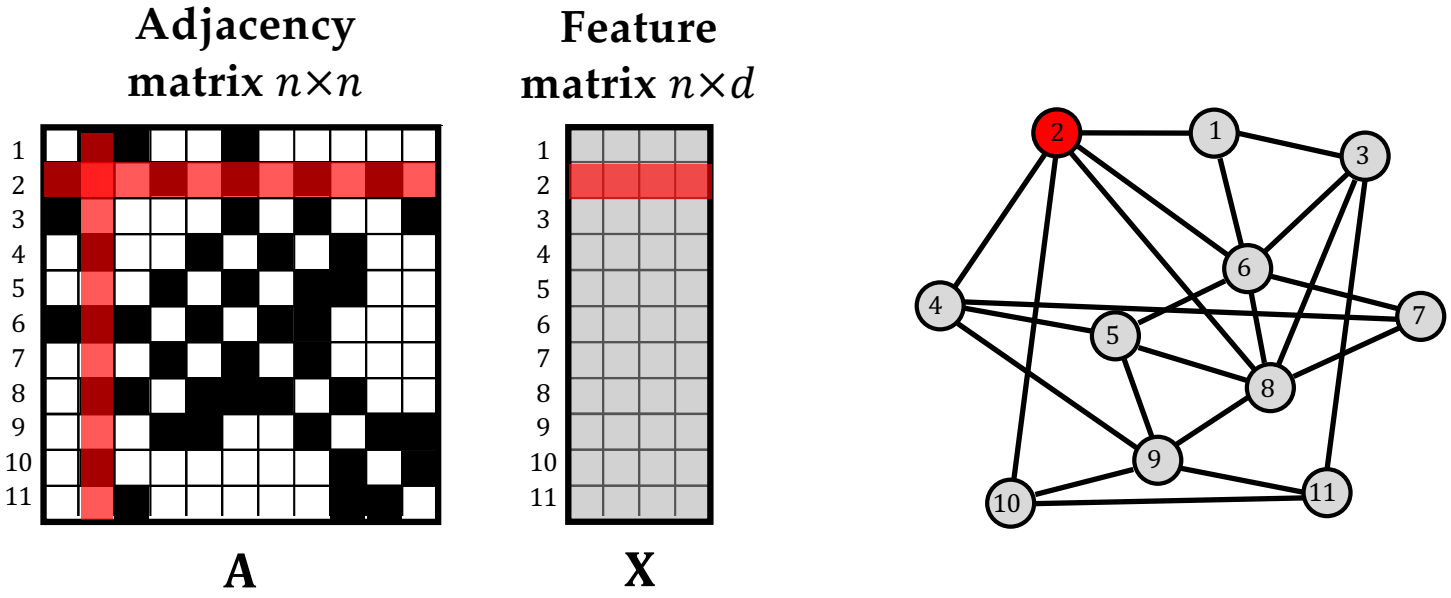
- Nodes indexed as $V = \{1, \ldots, n\}$ in *arbitrary order*
- Node features stacked row-wise in an $n \times d$ matrix $\mathbf{X}$

Representations of Graphs as matrices



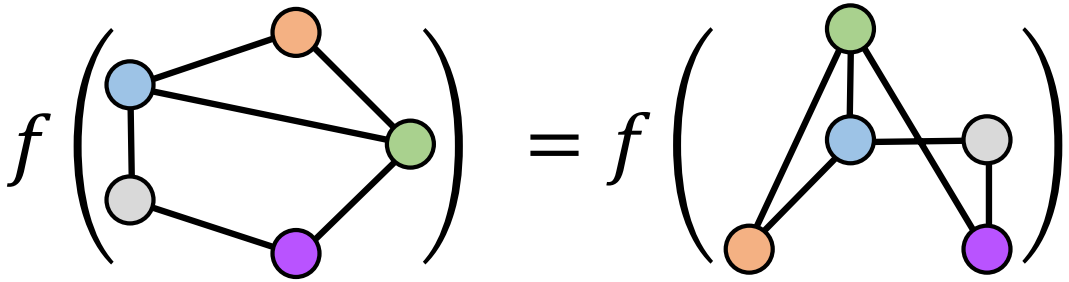
- Nodes indexed as $V = \{1, \ldots, n\}$ in *arbitrary order*
- Node features stacked row-wise in an $n \times d$ matrix **X**
- Graph structure represented by $n \times n$ **adjacency matrix A** with $a_{ij} = 1$ iff $i \sim j$

Representations of Graphs as matrices



The ordering of nodes is arbitrary!

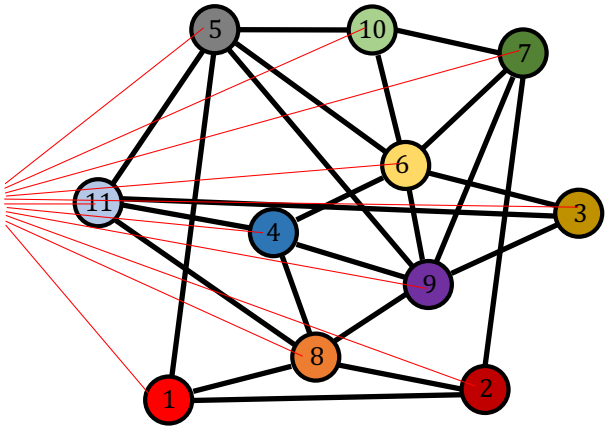
What do we want now with Graphs?



- The function now also depends on edges (adjacency $\mathbf{A}$) in addition to node features $\mathbf{X}$
- Permutation invariance: $f(\mathbf{PX}, \mathbf{PAP}^{\mathrm{T}}) = f(\mathbf{X}, \mathbf{A})$
- Permutation equivariance: $\mathbf{F}(\mathbf{PX}, \mathbf{PAP}^{\mathrm{T}}) = \mathbf{PF}(\mathbf{X}, \mathbf{A})$

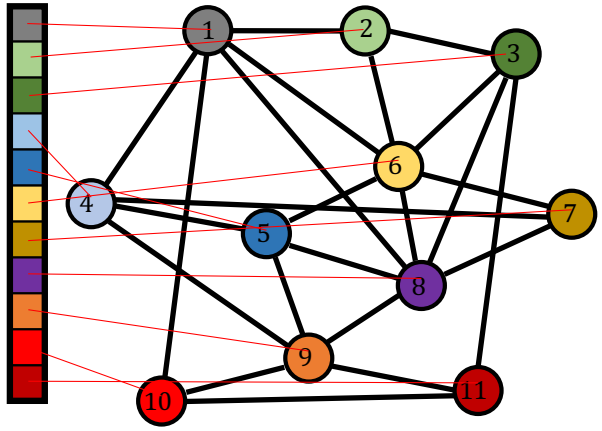
Invariant Graph Functions

$$f\left(\mathbf{PX}, \mathbf{PAP}^{\top}\right) = f(\mathbf{X}, \mathbf{A})$$

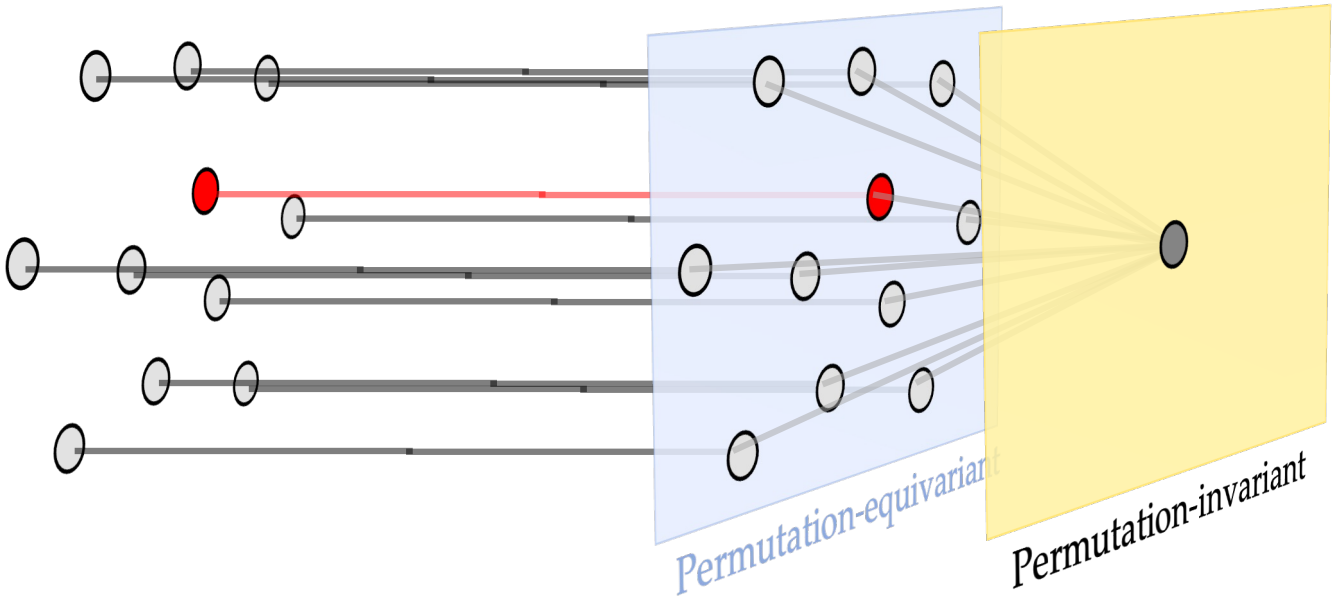


Equivariant Graph Functions

$$\mathbf{F}\left(\mathbf{PX}, \mathbf{PAP}^{\top}\right) = \mathbf{PF}(\mathbf{X}, \mathbf{A})$$

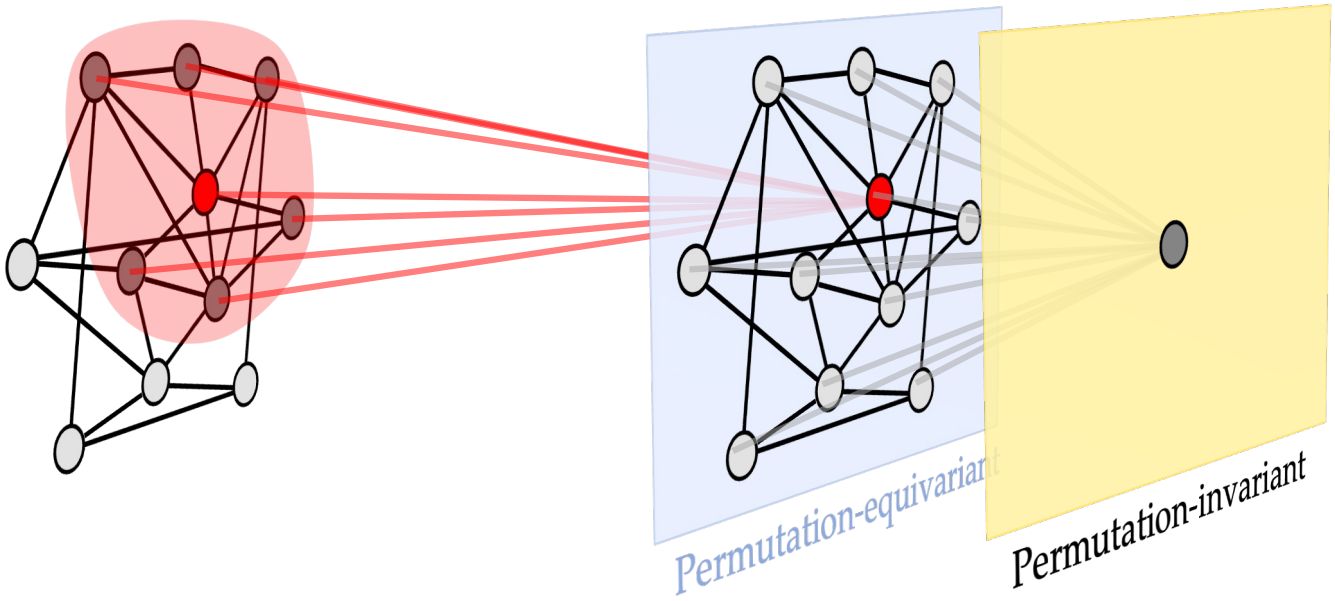


DeepSets



On sets, we can only process each node independently
(or all together)

Graph Neural Networks



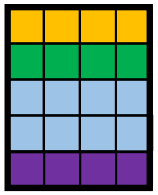
On graphs, we have a notion of a **neighbourhood** of ever node

$$\mathcal{N}_i = \{j \in V : i \sim j\}$$

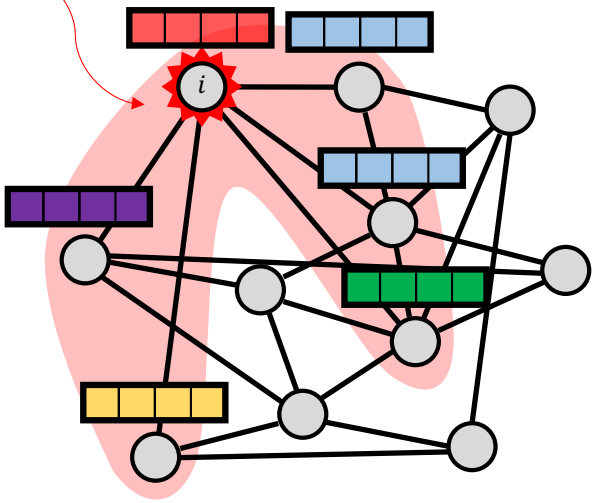
Neighbour Aggregation

neighbourhood
$\mathcal{N}_i = \{j : i \sim j\}$

multiset of
neighbour features



$$\mathbf{X}_{\mathcal{N}_i} = \{\!\{\mathbf{x}_{j \in \mathcal{N}_i}\}\!\}$$

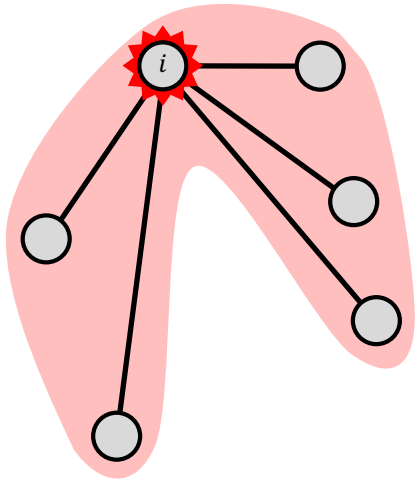


Neighbour Aggregation

local function

$$\phi\left(\begin{bmatrix}\mathbf{x}_i \\ \mathbf{x}_{\mathcal{N}_i}\end{bmatrix}\right)$$

permutation invariant



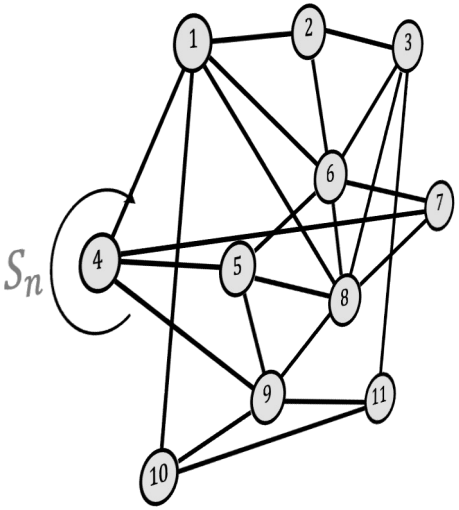
GNN Layer

$$\mathbf{F}(\mathbf{X}, \mathbf{A}) = \begin{pmatrix} - \phi(\mathbf{x}_1, \mathbf{X}_{\mathcal{N}_1}) - \\ \vdots \\ - \phi(\mathbf{x}_i, \mathbf{X}_{\mathcal{N}_i}) - \\ \vdots \\ - \phi(\mathbf{x}_n, \mathbf{X}_{\mathcal{N}_n}) - \end{pmatrix}$$

permutation equivariant

GNNs as an Instance of Geometric Deep Learning Blueprint

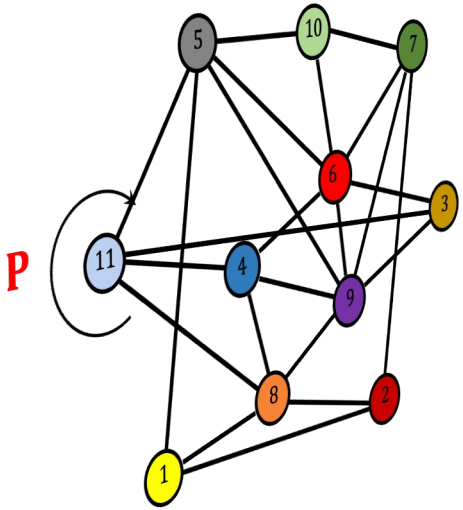
Graph $G = (V, E)$



Permutation group

$\pi \in S_n$

Node features $\mathcal{X}(G)$



Permutation matrix

$\mathbf{PX}, \mathbf{PAP}^\top$

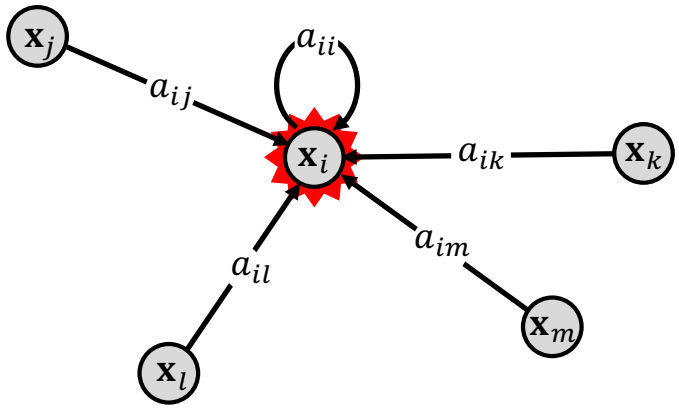
Functions $\mathcal{F}(\mathcal{X}(G))$

Message passing

$\mathbf{F}(\mathbf{PX}, \mathbf{PAP}^\top) = \mathbf{PF}(\mathbf{X}, \mathbf{A})$

GNN FLAVOURS

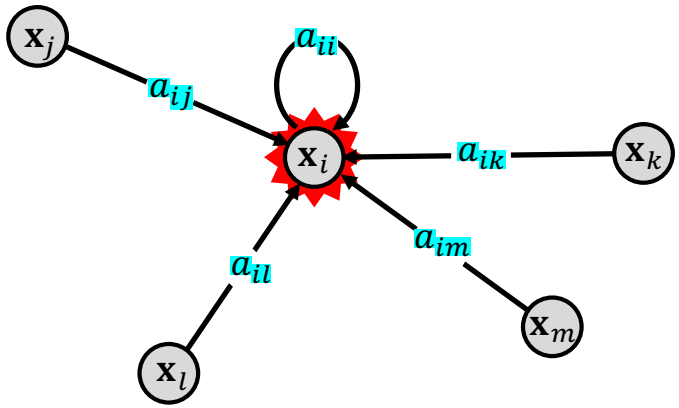
Convolutional GNNs



$$\phi\left(\mathbf{x}_i, \sum_{j \in \mathcal{N}_i} a_{ij}\psi(\mathbf{x}_j)\right)$$

Defferard et al. 2016; Kipf, Welling 2016 (GCN)

Convolutional GNNs

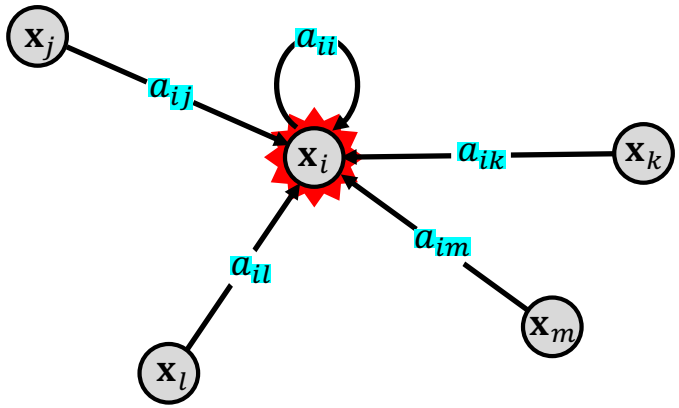


$$\phi\left(\mathbf{x}_i, \sum_{j \in \mathcal{N}_i} a_{ij} \psi(\mathbf{x}_j)\right)$$

graph adjacency

Defferard et al. 2016; Kipf, Welling 2016 (GCN)

Convolutional GNNs



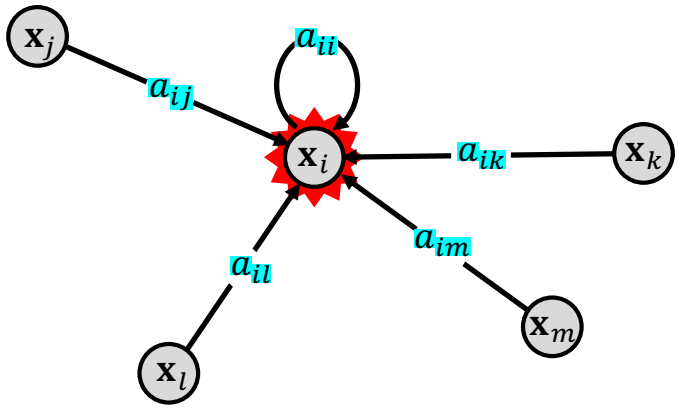
$$\phi\left(\mathbf{x}_i, \sum_{j \in \mathcal{N}_i} a_{ij} \psi(\mathbf{x}_j)\right)$$

graph adjacency

node-wise transformation

Defferard et al. 2016; Kipf, Welling 2016 (GCN)

Convolutional GNNs



$$\phi\left(\mathbf{x}_i, \sum_{j \in \mathcal{N}_i} a_{ij} \psi(\mathbf{x}_j)\right)$$

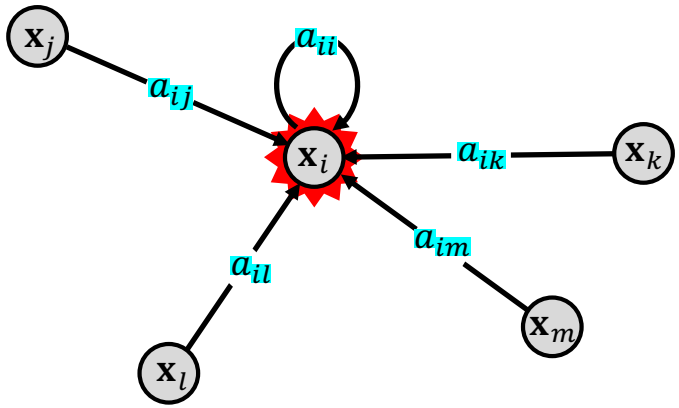
node feature

graph adjacency

node-wise transformation

Defferard et al. 2016; Kipf, Welling 2016 (GCN)

Convolutional GNNs



$$\phi\left(\mathbf{x}_i, \sum_{j \in \mathcal{N}_i} a_{ij} \mathbf{W} \mathbf{x}_j\right)$$

node feature

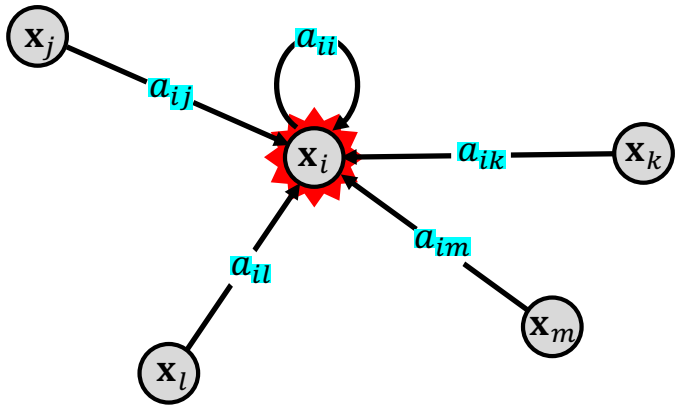
graph adjacency

node-wise linear transformation

Defferard et al. 2016; Kipf, Welling 2016 (GCN)

Convolutional GNNs

- Simplest GNN
- Highly scalable
- Industrial use cases
- Folklore: works only on homophilic graphs



$$\mathbf{X} \leftarrow \sigma(\mathbf{AXW})$$

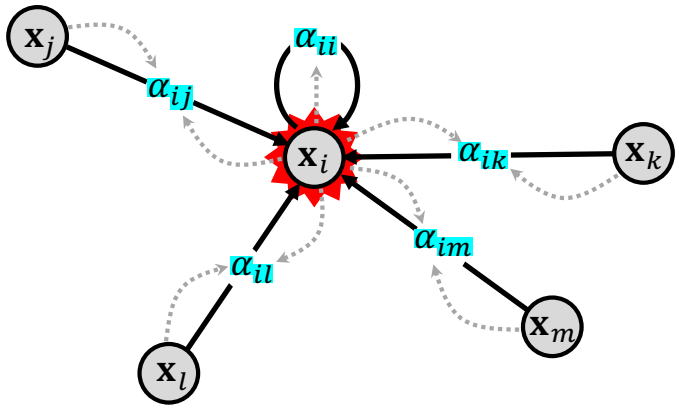
nonlinear activation

diffusion $n \times n$

channel mixing $d \times d$

Defferard et al. 2016; Kipf, Welling 2016 (GCN)
Rossi, Frasca et B 2020 (SIGN); Ying et al. 2018 (PinSAGE)

Attentional GNNs

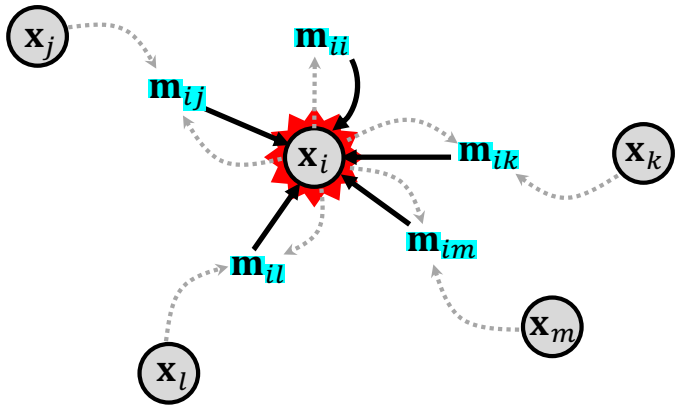


$$\phi\left(\mathbf{x}_i, \sum_{j \in \mathcal{N}_i} \alpha_{ij}(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$

learnable attention weights

Monti et B 2017; Veličković et al. 2018 (GAT)

Message-Passing GNNs

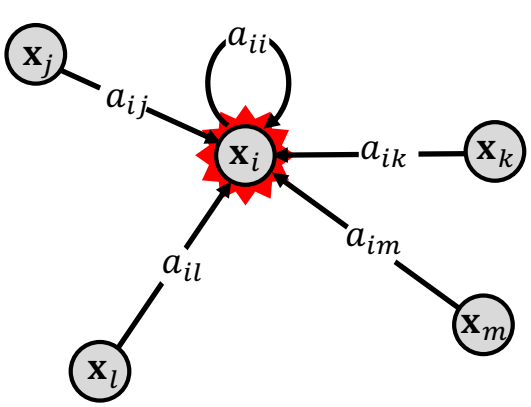


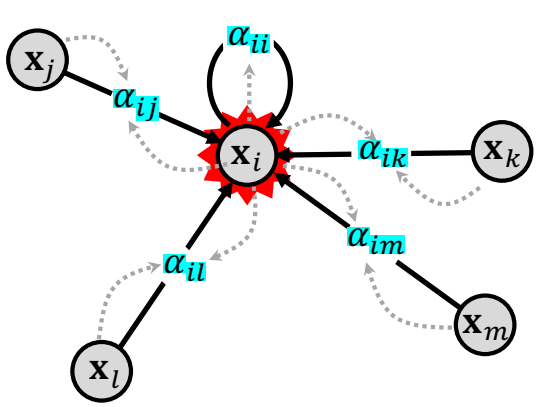
$$\phi\left(\mathbf{x}_i, \sum_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

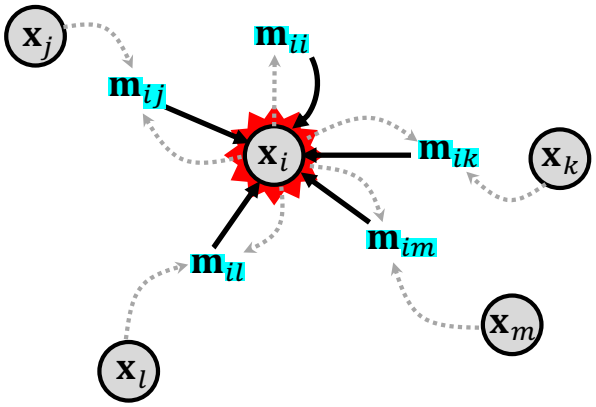
message from node j to node i

Gilmer et al. 2017 (MPNN); Battaglia et al. 2018 (Graph Networks)
Wang et B, Solomon 2018 (edgeconv)

Flavours of GNNs



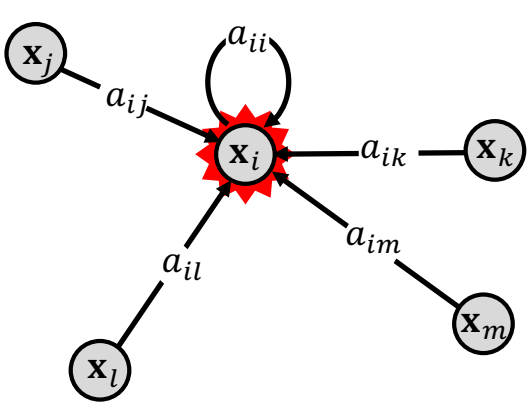


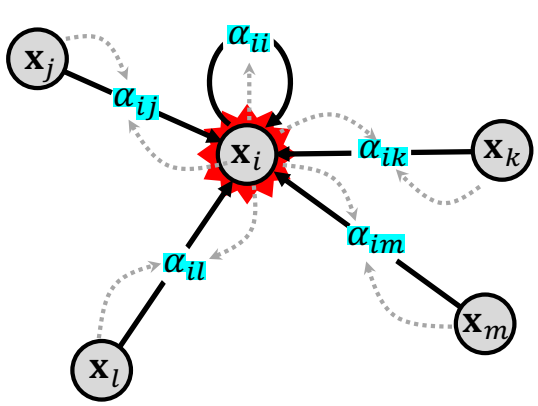


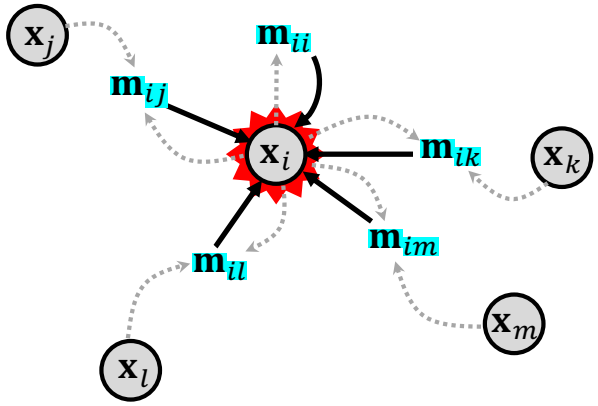
Convolutional **Attentional** **Generic Message Passing**

$$\phi\left(\mathbf{x}_i, \sum_{j \in \mathcal{N}_i} a_{ij}\psi(\mathbf{x}_j)\right) \subseteq \phi\left(\mathbf{x}_i, \sum_{j \in \mathcal{N}_i} \alpha_{ij}(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right) \subseteq \phi\left(\mathbf{x}_i, \sum_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

Flavours of GNNs





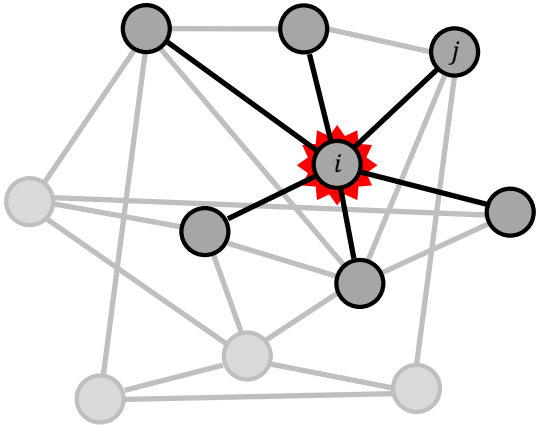


Convolutional **Attentional** **Generic Message Passing**

$$\mathbf{AX} \subseteq \mathbf{A}(\mathbf{X})\mathbf{X} \subseteq \mathcal{A}(\mathbf{X})$$

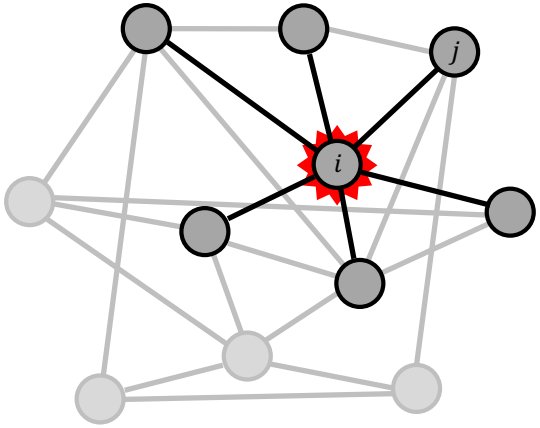
MAIN INGREDIENTS OF AN MPNN

Main Ingredients of an MPNN



$$\mathbf{x}_i \leftarrow \phi\left(\mathbf{x}_i, \square_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

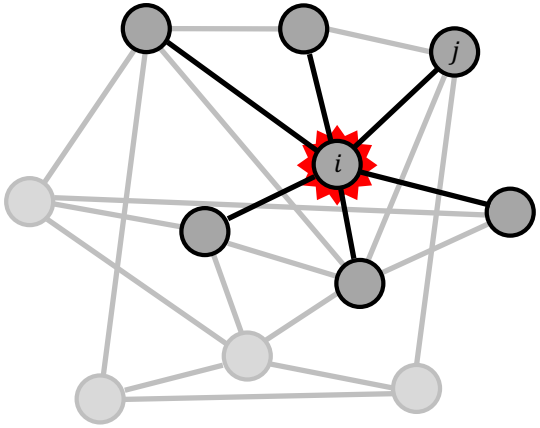
Main Ingredients of an MPNN



$$\mathbf{x}_i \leftarrow \phi\left(\mathbf{x}_i, \square_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

Message passing function

Main Ingredients of an MPNN

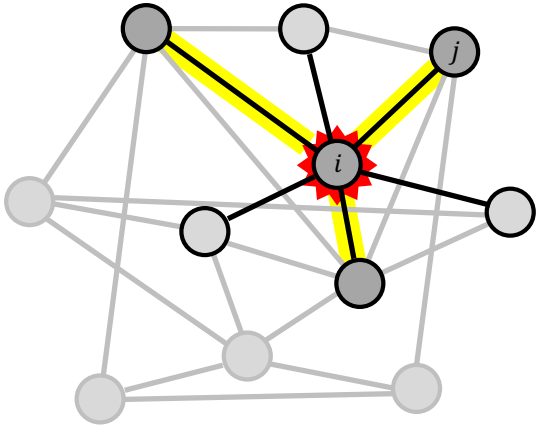


$$\mathbf{x}_i \leftarrow \phi\left(\mathbf{x}_i, \square_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

Message passing function

Aggregation operator

Main Ingredients of an MPNN



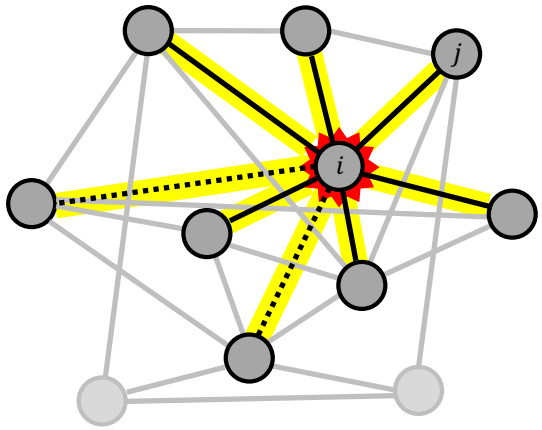
$$\mathbf{x}_i \leftarrow \phi\left(\mathbf{x}_i, \square_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

Message passing function

Aggregation operator

Computational graph

Main Ingredients of an MPNN



$$\mathbf{x}_i \leftarrow \phi\left(\mathbf{x}_i, \square_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

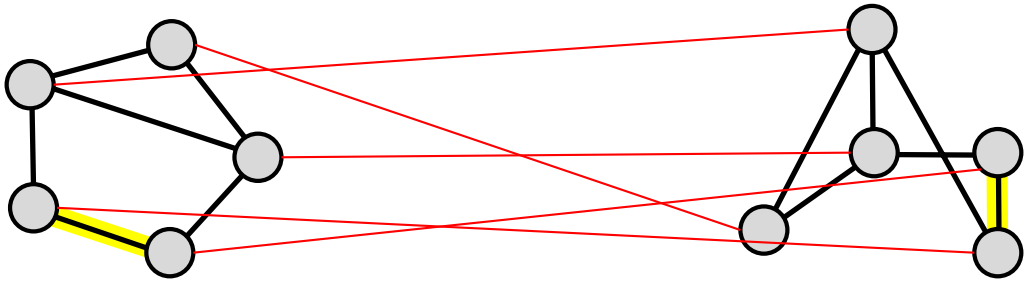
Message passing function

Aggregation operator

Computational graph

EXPRESSIVE POWER OF GNNS

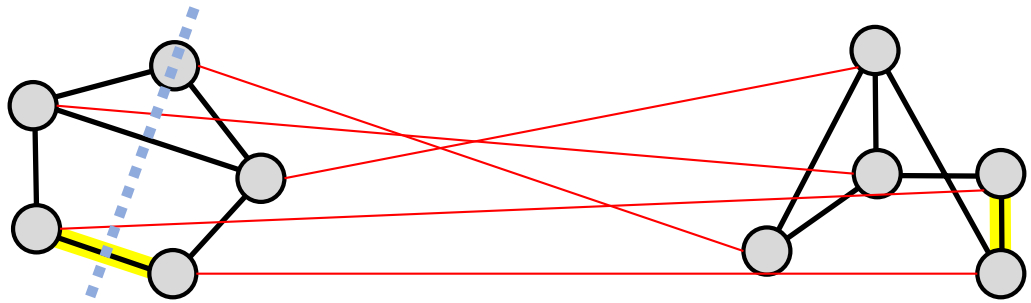
Graph isomorphism



$G = (V, E)$ $\qquad\qquad$ $G' = (V', E')$

- Two graphs $G = (V, E)$ and $G' = (V', E')$ are **isomorphic** if there exists an *edge-preserving bijection* $\varphi: V \to V'$ s.t. $u \sim v$ in G iff $\varphi(u) \sim \varphi(v)$ in G'

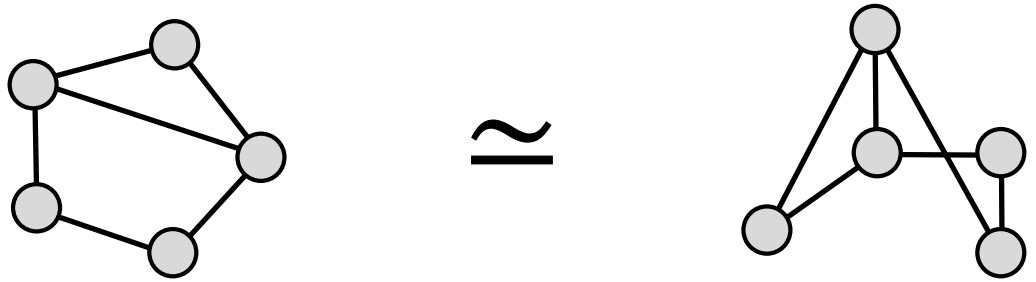
Graph isomorphism



- Two graphs $G = (V, E)$ and $G' = (V', E')$ are **isomorphic** if there exists an *edge-preserving bijection* $\varphi: V \to V'$ s.t. $u \sim v$ in G iff $\varphi(u) \sim \varphi(v)$ in G'

Note: φ is not unique where the graph has symmetries (edge-preserving automorphism)

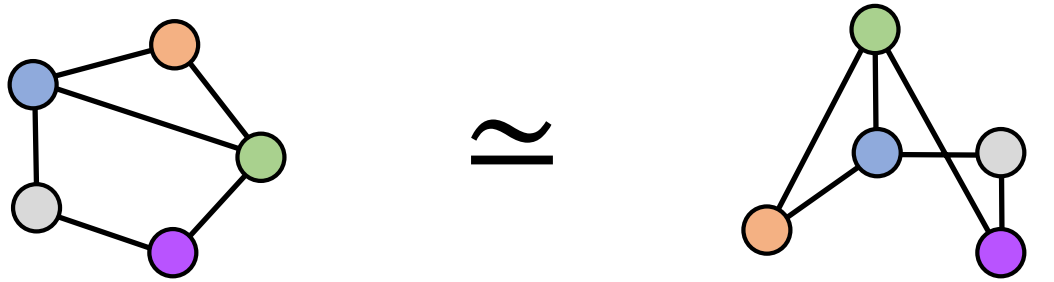
Graph isomorphism



$G = (V, E)$ $\qquad$ $G' = (V', E')$

- Two graphs $G = (V, E)$ and $G' = (V', E')$ are **isomorphic** if there exists an *edge-preserving bijection* $\varphi: V \to V'$ s.t. $u \sim v$ in G iff $\varphi(u) \sim \varphi(v)$ in G'
- A set of graphs isomorphic to each other is an isomorphism class
- Complexity of computing graph isomorphism is an open question (*"NP intermediate"*: not NP but also no polynomial time algorithm currently known)

Attributed graph isomorphism



$G = (V, E, \mathbf{X})$ $\quad\quad$ $G' = (V', E', \mathbf{X}')$

- Two node-attributed graphs $G = (V, E, \mathbf{X})$ and $G' = (V', E', \mathbf{X}')$ are **isomorphic** if there exists a *bijection* $\varphi: V \to V'$ s.t.
- *Structure preservation:* $\quad u \sim v$ in G iff $\varphi(u) \sim \varphi(v)$ in G'
- *Feature preservation:* $\quad \mathbf{x}_u = \mathbf{x}'_{\varphi(u)}$

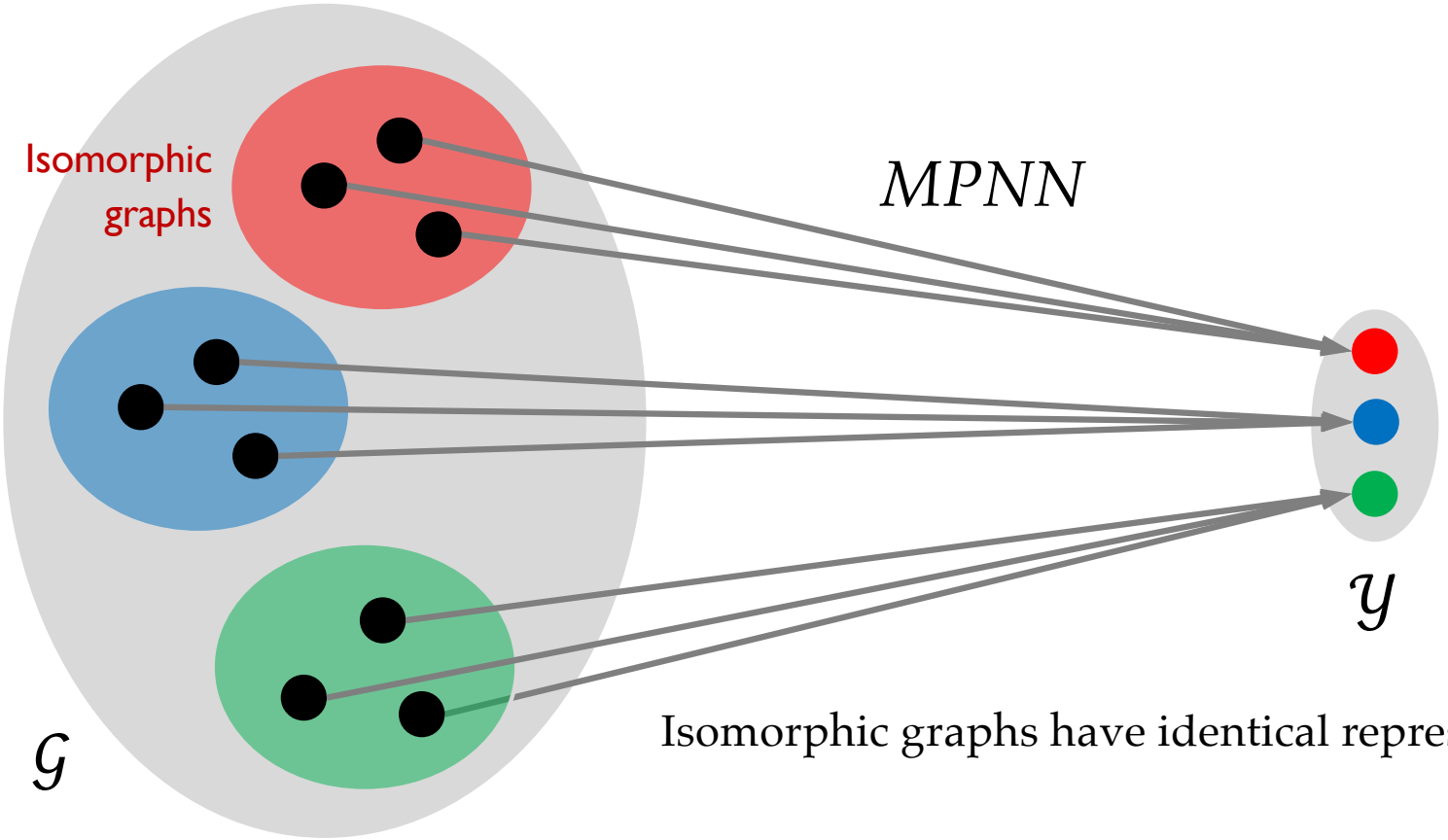
Universal Approximation on Graphs

Theorem: A class of functions is universally approximating permutation-invariant functions on graphs with finite node features iff it can discriminate graph isomorphisms.

Universal approximation on graphs is equivalent to graph isomorphism testing

Chen 2019

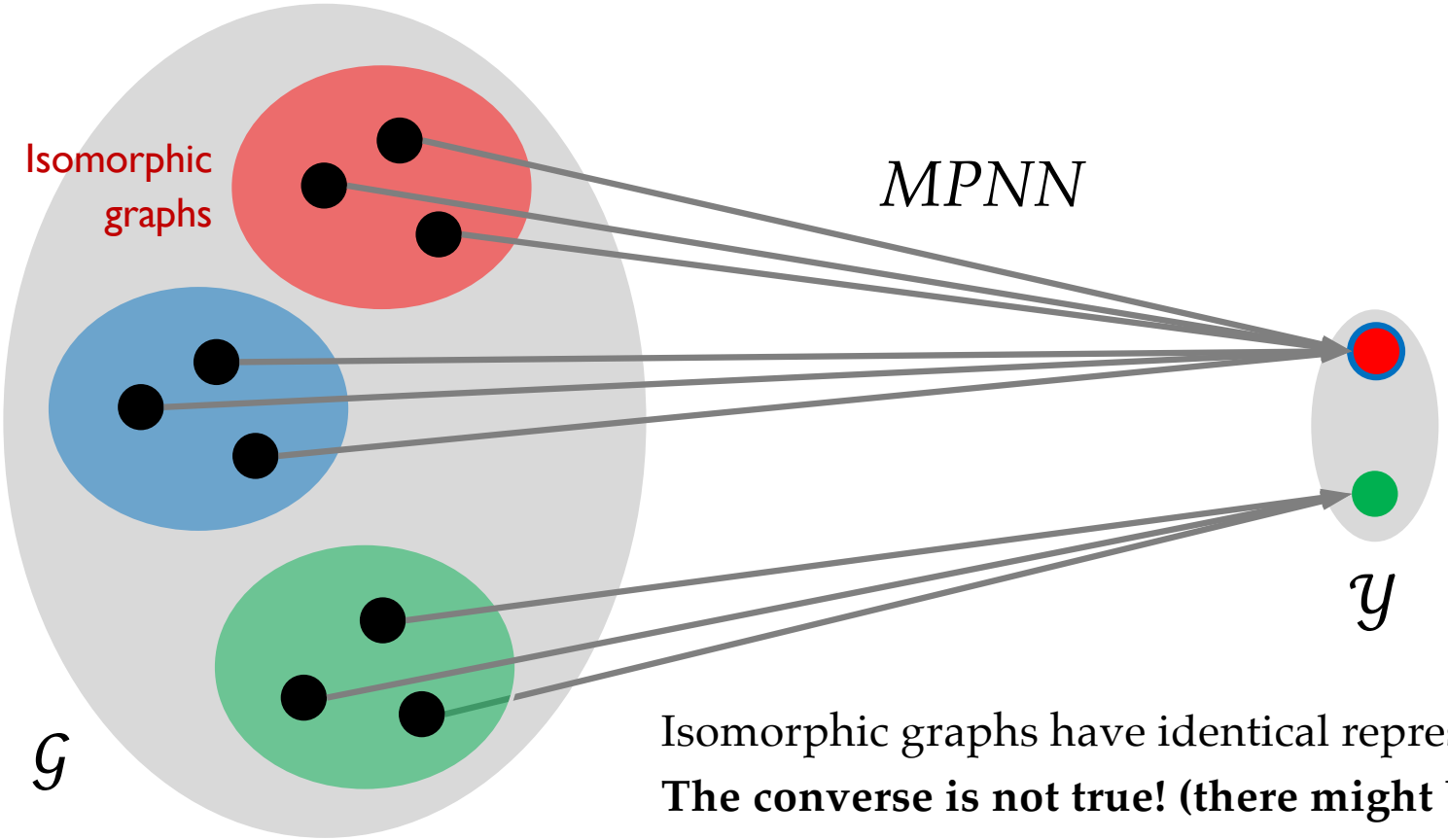
What graphs can MPNNs represent?



Isomorphic graphs have identical representations

Adapted from Bevilacqua et al.
(LoG Tutorial 2022)

What graphs can MPNNs represent?

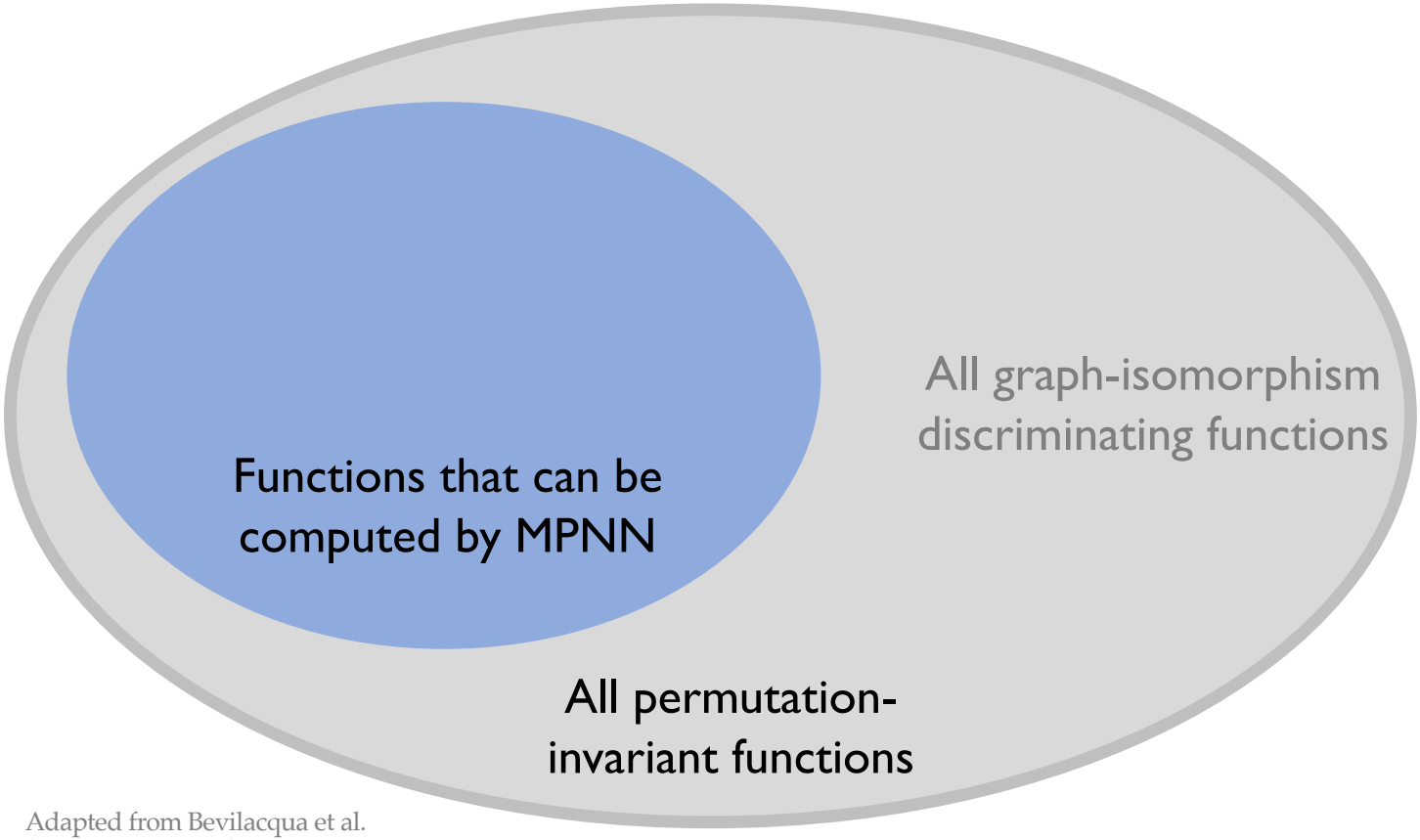


Isomorphic graphs have identical representation

The converse is not true! (there might be indistinguishable non-isomorphic graphs)

Adapted from Bevilacqua et al.
(LoG Tutorial 2022)

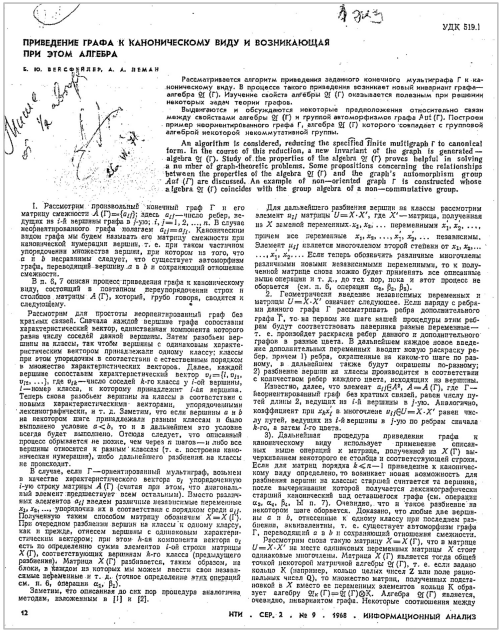
Expressive power of MPNNs



Adapted from Bevilacqua et al. (LoG Tutorial 2022)

WEISFEILER-LEHMAN TEST

Weisfeiler-Lehman Test



УДК 519.1

ПРИВЕДЕНИЕ ГРАФА К КАНОНИЧЕСКОМУ ВИДУ И ВОЗНИКАЮЩАЯ ПРИ ЭТОМ АЛГЕБРА

Б. Ю. ВЕЙСФЕЙЛЕР, А. А. ЛЕМАН

Рассматривается алгоритм приведения заданного конечного мультиграфа Γ к каноническому виду. В процессе такого приведения возникает новый инвариант графа — алгебра $\mathfrak{A}(\Gamma)$. Изучение свойств алгебры $\mathfrak{A}(\Gamma)$ оказывается полезным при решении некоторых задач теории графов.

Выдвигаются и обсуждаются некоторые предположения относительно связи между свойствами алгебры $\mathfrak{A}(\Gamma)$ и группой автоморфизмов графа $Aut(\Gamma)$. Построен пример неориентированного графа Γ, алгебра $\mathfrak{A}(\Gamma)$ которого совпадает с групповой алгеброй некоммутативной группы.

An algorithm is considered, reducing the specified finite multigraph Γ to canonical form. In the course of this reduction, a new invariant of the graph is generated — algebra $\mathfrak{A}(\Gamma)$. Study of the properties of the algebra $\mathfrak{A}(\Gamma)$ proves helpful in solving a no mber of graph-theoretic problems. Some propositions concerning the relationships between the properties of the algebra $\mathfrak{A}(\Gamma)$ and the graph's automorphism group $Aut(\Gamma)$ are discussed. An example of non-oriented graph Γ is constructed whose algebra $\mathfrak{A}(\Gamma)$ coincides with the group algebra of a non-commutative group.

1. Рассмотрим произвольный конечный граф Γ и его матрицу смежности $A(\Gamma)=\{a_{ij}\}$; здесь a_{ij} — число ребер, ведущих из i-й вершины графа в j-ую; $i, j=1, 2, \ldots, n$. В случае неориентированного графа полагаем $a_{ij}=a_{ji}$. Каноническим видом графа мы будем называть его матрицу смежности при канонической нумерации вершин, т. е. при таком частичном упорядочении множества вершин, при котором из того, что $a \neq b$ неравенству следует, что существует автоморфизм графа, переводящий вершину a в b и сохраняющий отношение смежности.

В п. 5, 7 описан процесс приведения графа к каноническому виду, состоящий в поэтапном перегруппировании строк и столбцов матрицы $A(\Gamma)$, который, грубо говоря, сводится к следующему.

Рассмотрим для простоты неориентированный граф без кратных связей. Сначала каждой вершине графа сопоставим характеристический вектор, единственная компонента которого равна числу соседей данной вершины. Затем разобьем вершины на классы, так чтобы вершины с одинаковым характеристическим вектором принадлежали одному классу: классы при этом упорядочим в соответствии с естественным порядком в множестве характеристических векторов. Далее, каждой вершине сопоставим характеристический вектор $v_i=(l, v_{i1}, v_{i2}, \ldots)$, где v_{ik} — число соседей k-го класса у i-ой вершины, l — номер класса, к которому принадлежит i-ая вершина. Теперь снова разобьем вершины на классы в соответствии с новыми характеристическими векторами, упорядочивании лексикографически, и т. д. Заметим, что если вершины a и b на некотором шаге принадлежали разным классам и было выполнено условие $a<b$, то и в дальнейшем это условие всегда будет выполнено. Отсюда следует, что описанный процесс обрывается не позже, чем через n шагов — и либо все вершины относятся к разным классам (т. е. построена каноническая нумерация), либо дальнейшего разбиения на классы не происходит.

В случае, если Γ — ориентированный мультиграф, возьмем в качестве характеристического вектора v_i упорядоченную i-ую строку матрицы $A(\Gamma)$ (считая при этом, что диагональный элемент предшествует всем остальным). Вместо различных элементов a_{ij} введем различные неизвестные переменные $x_1, x_2, \ldots$, упорядочив их в соответствии с порядком среди a_{ij}. Полученную таким способом матрицу обозначим $X=X(\Gamma)$. При очередном разбиении вершин на классы к одному классу, как и прежде, отнесем вершины с одинаковым характеристическим вектором; при этом k-ая компонента вектора v_i есть по определению сумма элементов i-ой строки матрицы $X(\Gamma)$, соответствующих вершинам k-го класса (предыдущего разбиения). Матрица $X(\Gamma)$ разбивается, таким образом, на блоки, в каждом из которых мы можем ввести свои независимые переменные и т. д. (точное определение этих операций см. п. 6, формулы α_3, β_3).

Заметим, что описанная до сих пор процедура аналогична методам, изложенным в [1] и [2].

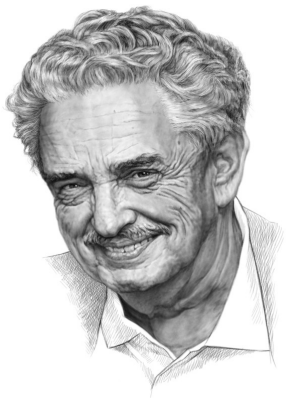
Для дальнейшего разбиение вершин на классы рассмотрим элемент u_{ij} матрицы $U=X-X'$, где X' — матрица, получаемая из X заменой переменных $x_2, x_3, \ldots$ переменными $x_1, x_2, \ldots$, причем все переменные $x_1, x_2, \ldots, x_s, x_{s+1}, \ldots$ независимы. Элемент u_{ij} является многочленом второй степени от $x_1, x_2, \ldots, x_1, x_2, \ldots$. Если теперь обозначить различные многочлены различными новыми независимыми переменными, то к полученной матрице снова можно будет применять все описанные выше операции и т. д., до тех пор, пока в этот процесс не обогатится (см. п. 6, операции α_4, β_4, β_5).

2. Геометрическое введение независимых переменных в матрицы $U=X-X'$ означает следующее. Если вернуться к ребрам данного графа Γ, рассматривать ребра дополнительного графа $\overline{\Gamma}$, то на первом же шаге нашей процедуры этим ребрам будут соответствовать различные разные переменные — т. е. произойдет раскраска ребер данного и дополнительного графов в разные цвета. В дальнейшем каждое новое введение дополнительных переменных вводит новую раскраску ребер, причем 1) ребра, окрашенные на каком-то шаге по разному, в дальнейшем также будут окрашены по-разному; 2) разбиение вершин на классы производится в соответствии с количеством ребер каждого цвета, исходящих из вершины.

Известно, далее, что элемент $a_{ij} \in A^k$, $A=A(\Gamma)$, где Γ — неориентированный граф без кратных связей, равен числу путей длины k, ведущих из i-й вершины в j-ую. Аналогичный, коэффициент при $x_s x_t$ в многочлене $a_{ij} \in U=X \cdot X'$ равен числу путей, ведущих из i-й вершины в j-ую по ребрам сначала s-го, а затем t-го цвета.

3). Дальнейшая процедура приведения графа к каноническому виду использует применение описанных выше операций к матрице, полученной из $X(\Gamma)$ вычеркиванием некоторого ее столбца и соответствующей строки. Если для матриц порядка $k \leqslant n-1$ приведение к каноническому виду определено, то возникает новая возможность для разбиения вершин на классы: старшей считается та вершина, после вычеркивания которой получается лексикографически старший канонический над оставшегося графа (см. операции α_5, α_6, β_6, β_7, δ п. 7). Очевидно, что в таком разбиении на некотором шаге обогатится. Доказано, что любые две вершины a и b, отнесенные к одному классу при последнем разбиении, эквивалентны, т. е. существует автоморфизм графа Γ, переводящий a в b и сохраняющий отношение смежности.

Рассмотрим снова такую матрицу $X=X(\Gamma)$, что в матрице $U=X-X'$ на месте одинаковых переменных матрицы X стоит одинаковые многочлены. Матрица $X(\Gamma)$ является тогда общей точкой некоторой матричной алгебры $\mathfrak{A}(\Gamma)$, т. е. если задано кольцо K (например, кольцо целых чисел Z или поле рациональных чисел Q), то множество матриц, полученных подстановкой в X вместо ее переменных элементов кольца K образует алгебру $\mathfrak{A}_K(\Gamma)=\mathfrak{A}(\Gamma) \otimes K$. Алгебра $\mathfrak{A}(\Gamma)$ является, очевидно, инвариантом графа. Некоторые соотношения между

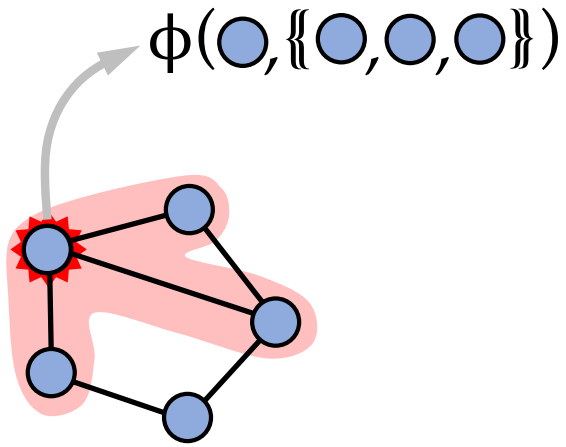


A. Le(h)man

B. Weisfeiler

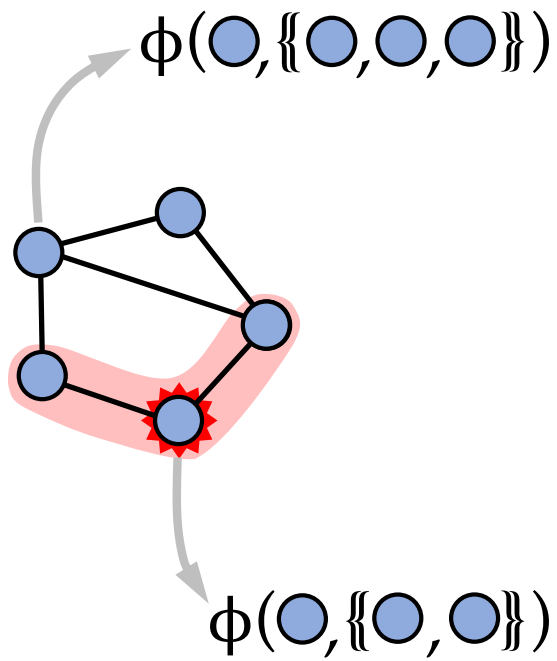
Weisfeiler, Lehman 1968

Weisfeiler-Lehman Test



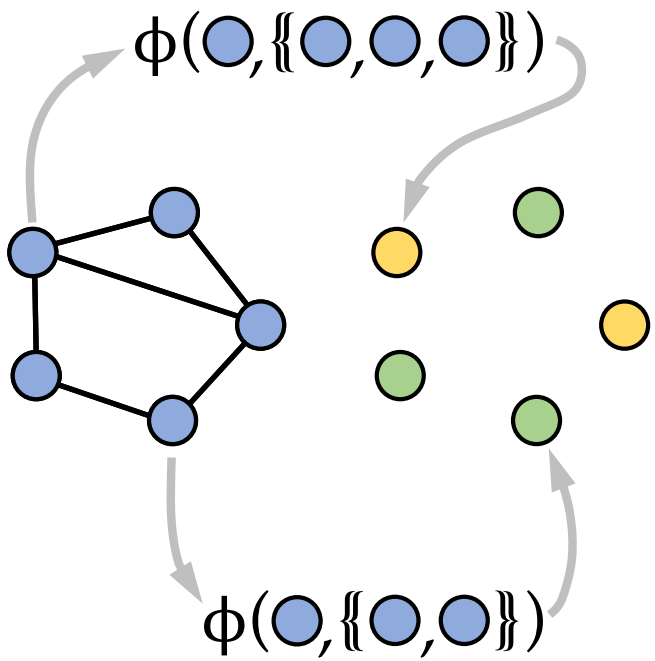
Weisfeiler, Lehman 1968

Weisfeiler-Lehman Test



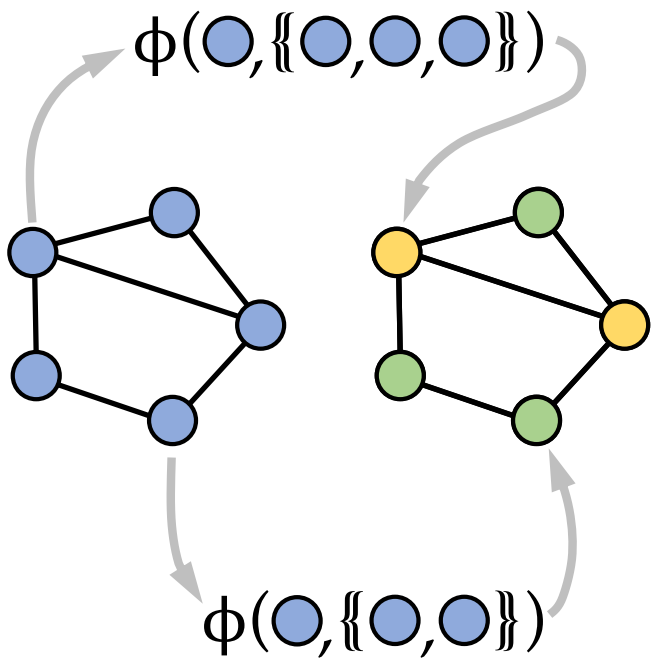
Weisfeiler, Lehman 1968

Weisfeiler-Lehman Test



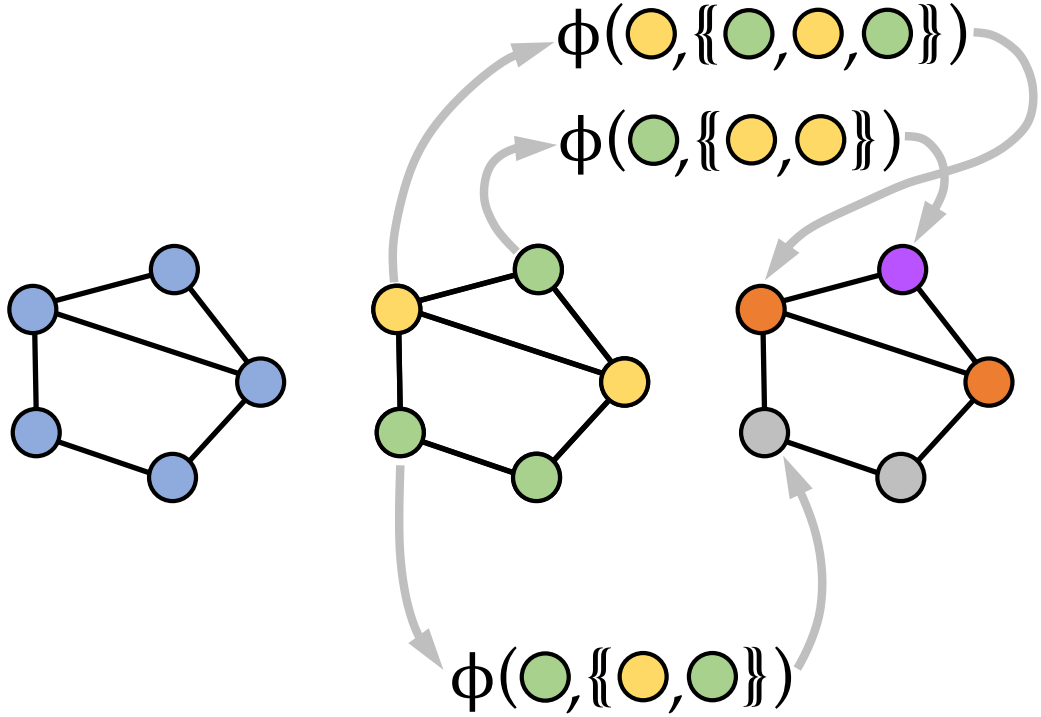
Weisfeiler, Lehman 1968

Weisfeiler-Lehman Test



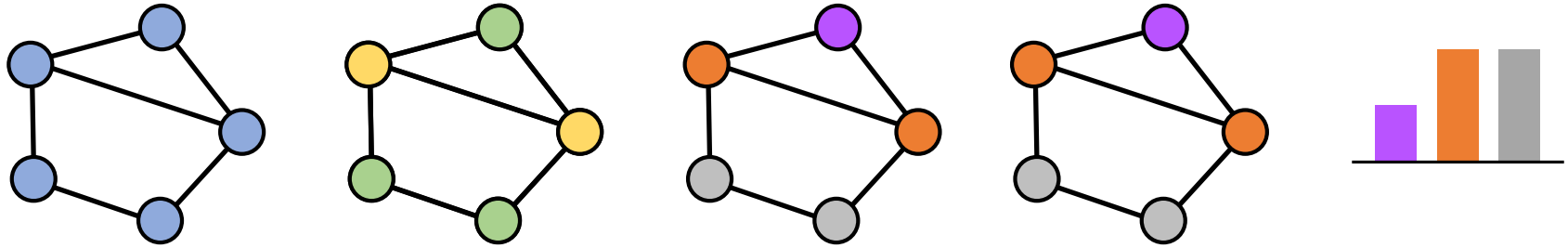
Weisfeiler, Lehman 1968

Weisfeiler-Lehman Test



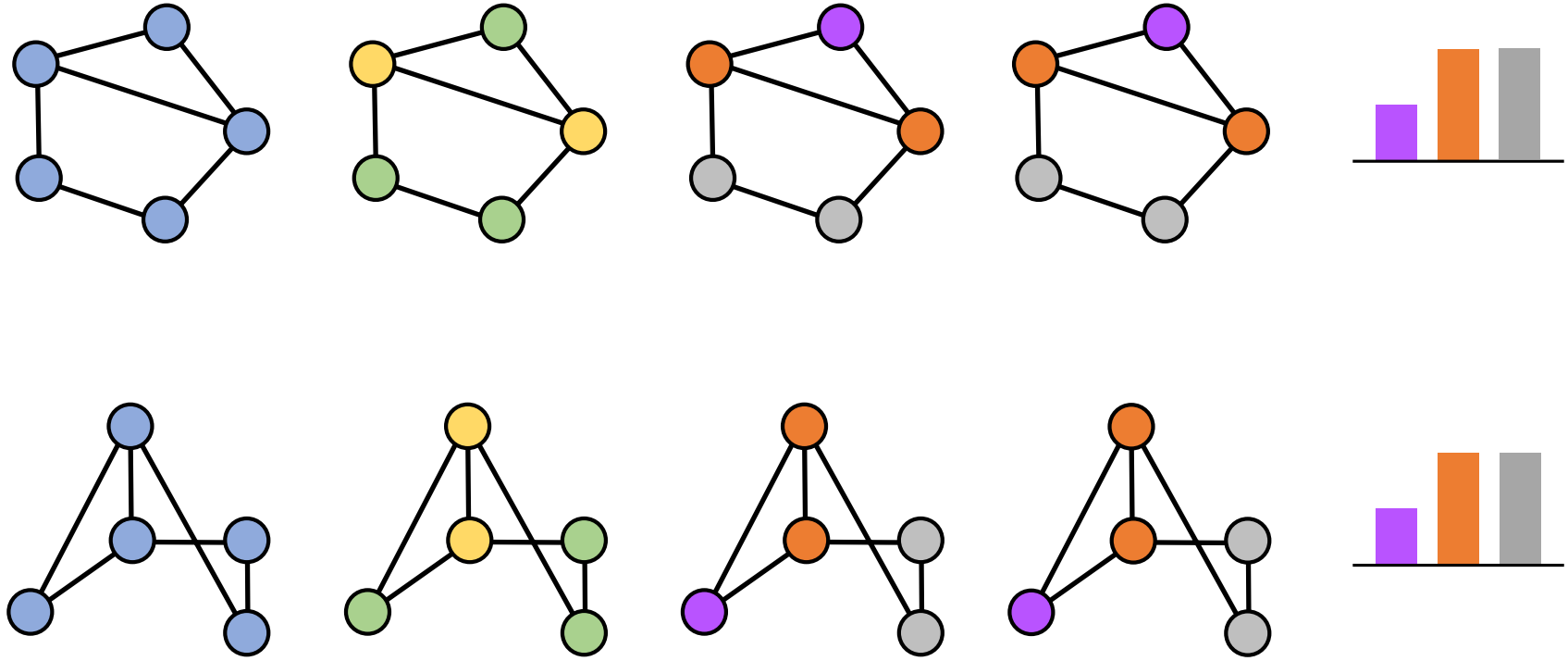
Weisfeiler, Lehman 1968

Weisfeiler-Lehman Test



Weisfeiler, Lehman 1968

Weisfeiler-Lehman Test

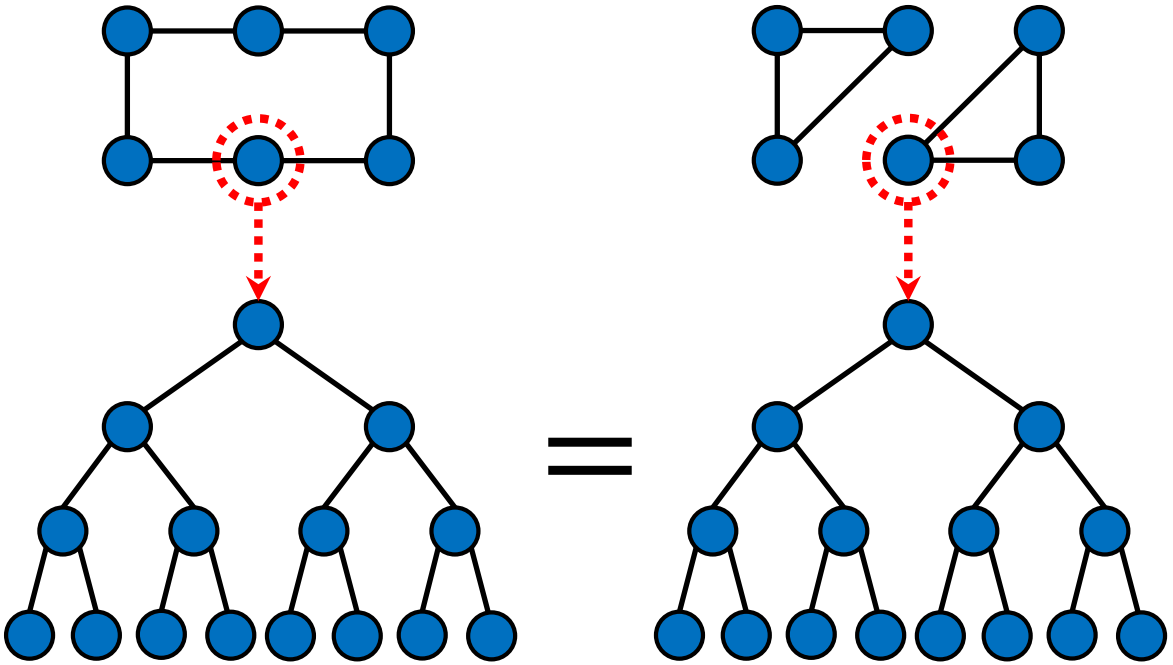


Weisfeiler, Lehman 1968

non-isomorphic graphs that are WL-equivalent

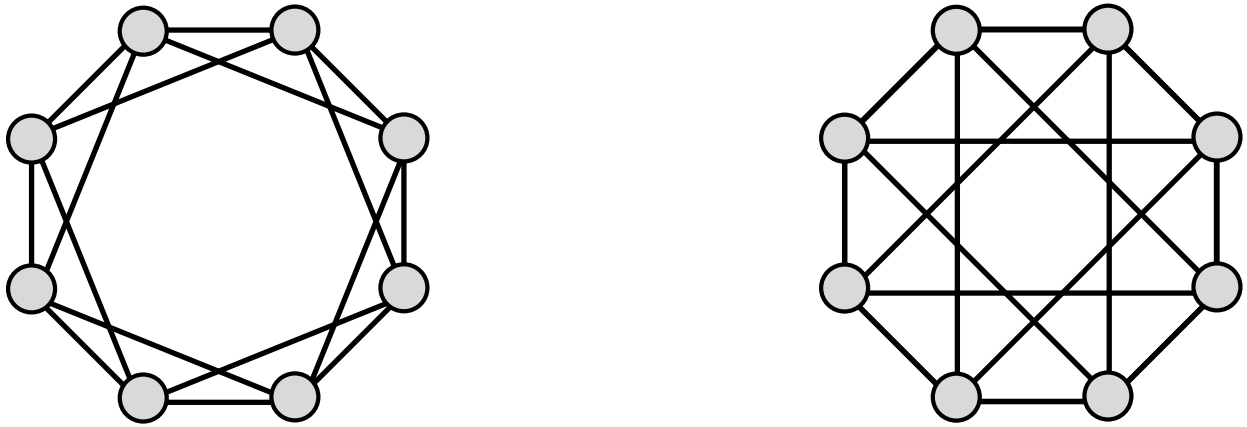
Necessary but insufficient condition for graph isomorphism!

What does WL test see?



What WL cannot test?

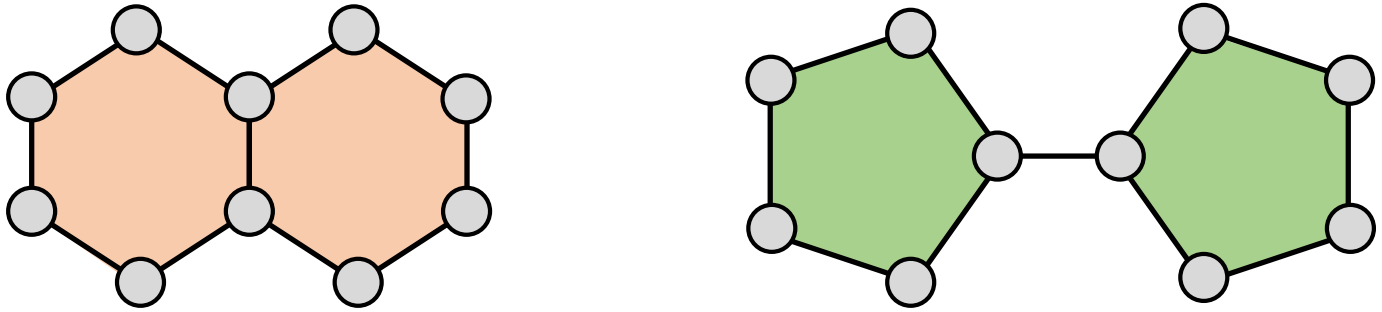
Example of non-isomorphic graphs that cannot be distinguished by Weisfeiler-Lehman test (outputs “possibly isomorphic”)



r-regular graphs (deg=r at every node) with the same number of nodes

What WL cannot test?

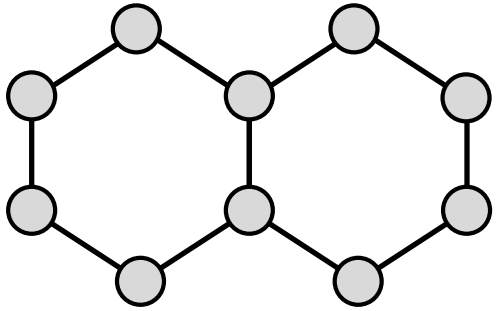
Example of non-isomorphic graphs that cannot be distinguished by Weisfeiler-Lehman test (outputs "possibly isomorphic")



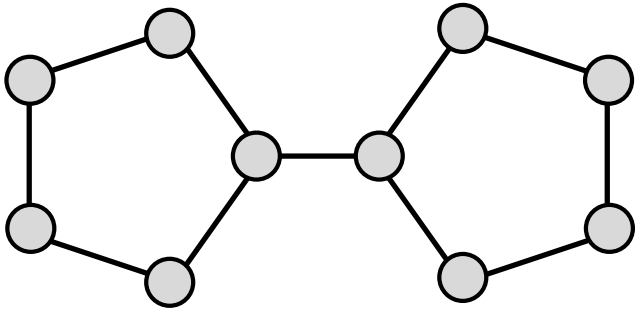
Any induced connected pattern with ≥ 3 *nodes (triangles, cycles, etc.)*

What WL cannot test?

Example of non-isomorphic graphs that cannot be distinguished by Weisfeiler-Lehman test (outputs “possibly isomorphic”)



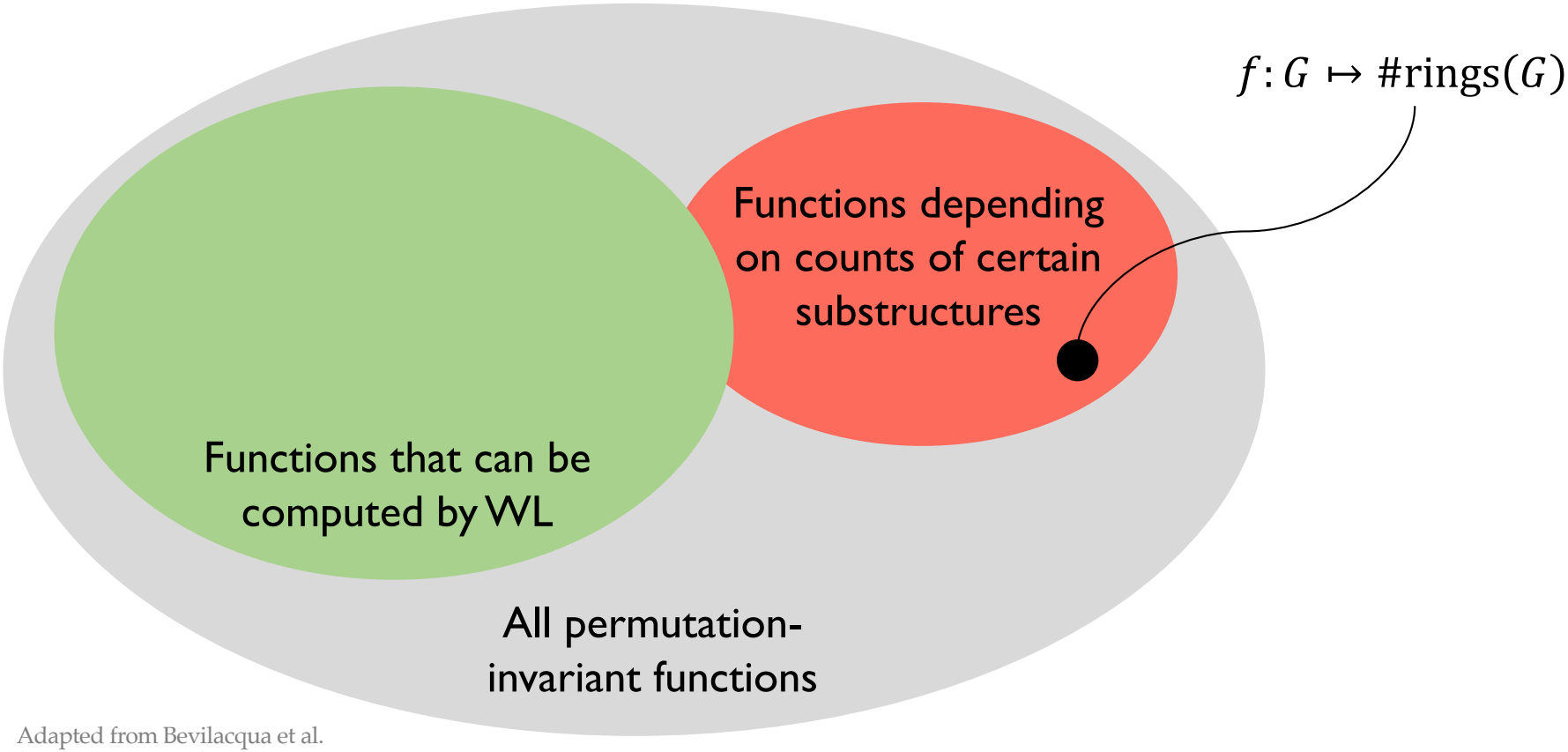
decalin



bicyclopnetyl

Important implications e.g. in chemistry!

Expressive power of Weisfeiler-Lehman

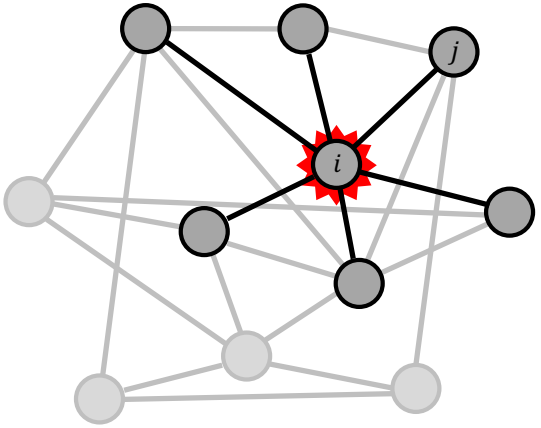


Adapted from Bevilacqua et al. (LoG Tutorial 2022)

MPNNs vs WL

- **WL-test:** $\mathbf{x}_i \leftarrow \phi\left(\mathbf{x}_i, \{\!\{\mathbf{x}_j : j \in \mathcal{N}_i\}\!\}\right)$
  where ϕ is *injective* (hash function)

- **MPNN:** $\mathbf{x}_i \leftarrow \phi\left(\mathbf{x}_i, \square_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$

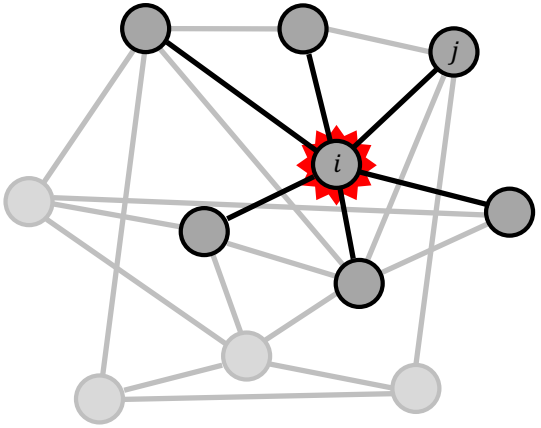


MPNN expressive power is *upper-bounded* by the Weisfeiler-Lehman test

MPNNs vs WL

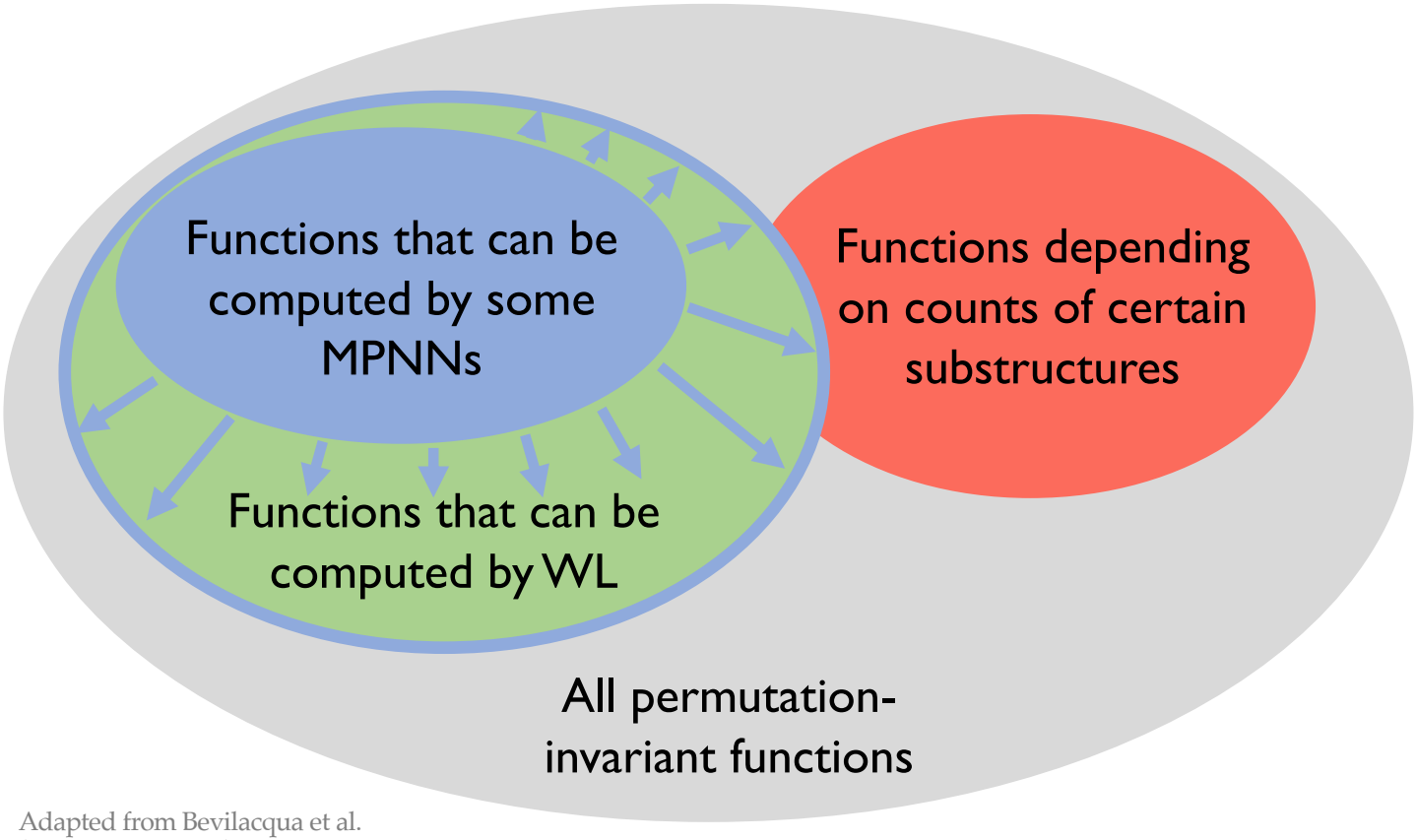
- **WL-test:** $\mathbf{x}_i \leftarrow \phi\left(\mathbf{x}_i, \{\!\{\mathbf{x}_j : j \in \mathcal{N}_i\}\!\}\right)$
  where ϕ is *injective* (hash function)

- **MPNN:** $\mathbf{x}_i \leftarrow \phi\left(\mathbf{x}_i, \square_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$



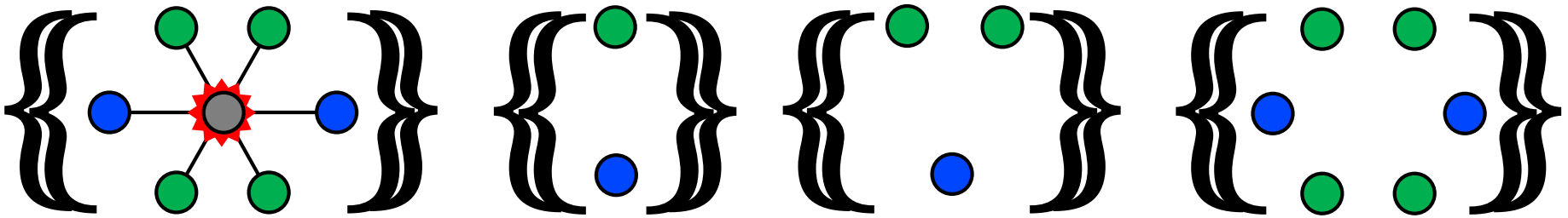
When is MPNN as expressive as the Weisfeiler-Lehman test?

Expressive power of MPNNs



Adapted from Bevilacqua et al. (LoG Tutorial 2022)

Are all Aggregators the same?

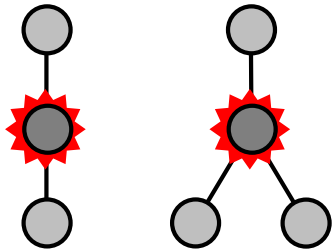


Input **max** **mean** **sum**

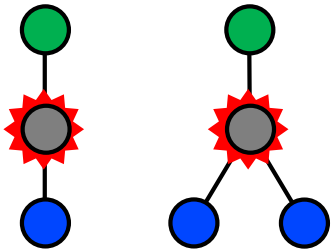
"skeleton"
of the multiset

Distribution
of the multiset

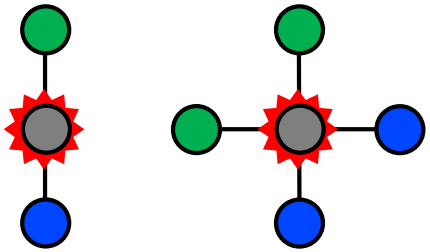
Are all Aggregators the same?



mean and **max**
fail to distinguish



max
fails to distinguish



mean and **max**
fail to distinguish

Graph Isomorphism Network (GIN)

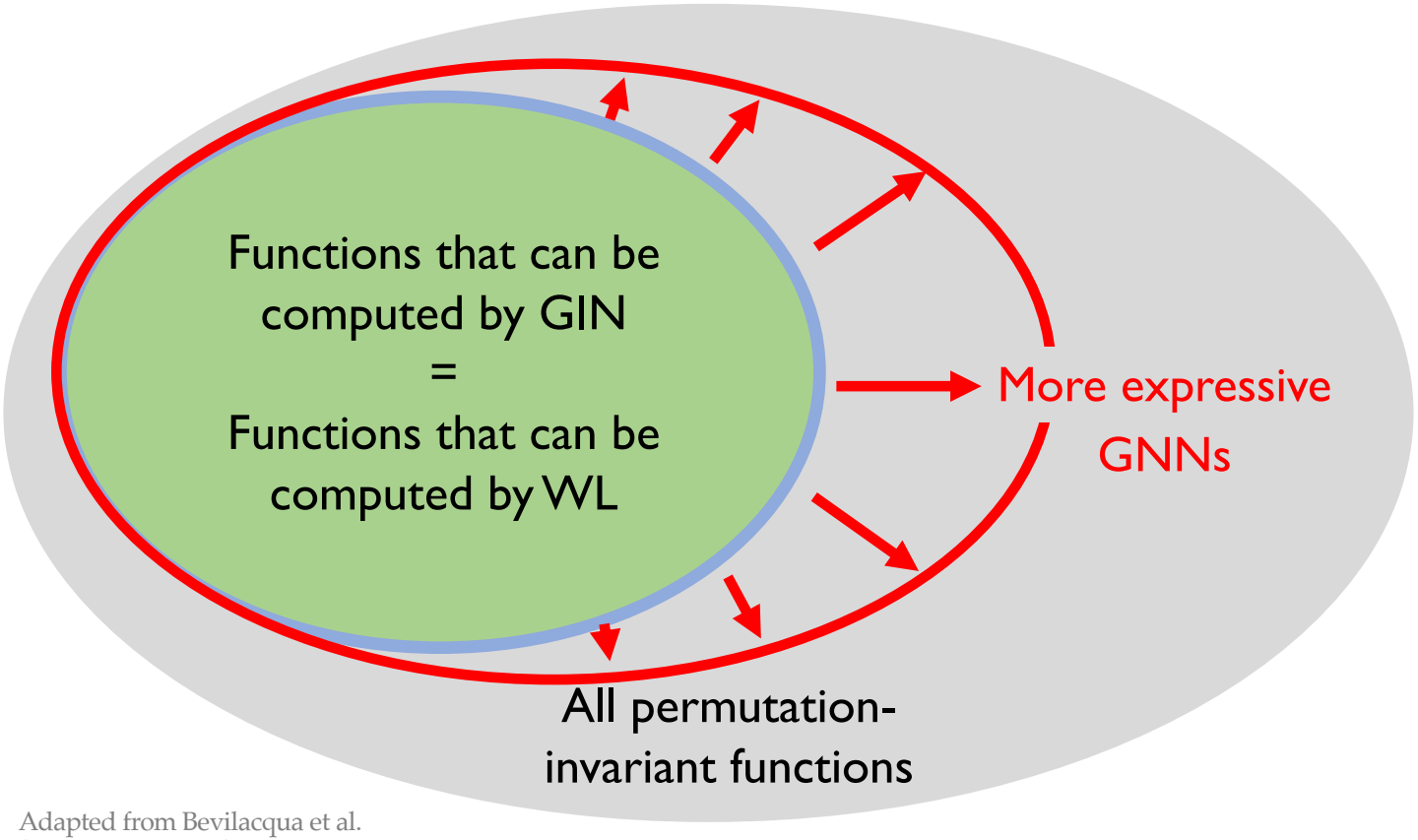
Theorem: Assume graph node features are from a countable set. Then, an MPNN with with injective aggregator $\square$, update function ϕ, and graph-wise readout function, is as powerful as the Weisfeiler-Lehman test.

- Assumption of discrete countable features (often not the case in practice)
- Proof similar to DeepSets, with the difference that we now deal with *multisets*, where popular injective set functions such as mean are not injective anymore
- GIN: uses an injective multiset function of the form

$$\text{MLP}\left((1+\epsilon)\mathbf{x}_i + \sum_{j \in \mathcal{N}_i} \mathbf{x}_j\right)$$

Xu 2019; (Morris 2019)

Expressive power of GIN ("best MPNN")



Adapted from Bevilacqua et al. (LoG Tutorial 2022)

What to do with continuous features?

- Many practical applications rely on assumption of continuous (uncountable) features
- Most of the results we have seen (DeepSets, GINs, etc.) do not work in this setting!

> **Theorem:** In order to discriminate between real multisets of size n, at least n aggregators are needed.

Proof: relying on Borsuk-Ulam Theorem (continuous function from $\mathbb{S}^n$ to $\mathbb{R}^n$ maps some pairs of antipodal points to the same point)

Corso et al. 2020

Principal Neighbourhood Aggregation (PNA)

- Use a *combination* of multiple aggregators defined as moments of neighbour features

$$\square = \begin{bmatrix} I \\ S(D, \alpha = 1) \\ S(D, \alpha = -1) \end{bmatrix} \otimes \begin{bmatrix} \mu \\ \sigma \\ \max \\ \min \end{bmatrix}$$

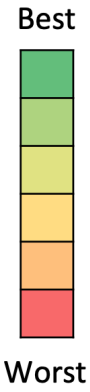
  (*scalers* $S(d, \alpha) = \left(\frac{\log(d+1)}{\delta}\right)^{\alpha}$ emphasize hub nodes and allow for aggregators like sum)

- Empirically good performance

Corso et al. 2020

Principal Neighbourhood Aggregation (PNA)

		Nodes tasks			Graph tasks		
Model	**Average score**	**1**	**2**	**3**	**4**	**5**	**6**
PNA	**-3.13**	**-2.89**	**-2.89**	**-3.77**	**-2.61**	**-3.04**	**-3.57**
PNA (no scalers)	-2.77	-2.54	-2.42	-2.94	**-2.61**	-2.82	-3.29
MPNN (max)	-2.53	-2.36	-2.16	-2.59	-2.54	-2.67	-2.87
MPNN (sum)	-2.50	-2.33	-2.26	-2.37	-1.82	-2.69	-3.52
GAT	-2.26	-2.34	-2.09	-1.60	-2.44	-2.40	-2.70
GCN	-2.04	-2.16	-1.89	-1.60	-1.69	-2.14	-2.79
GIN	-1.99	-2.00	-1.90	-1.60	-1.61	-2.17	-2.66
Baseline	-1.38	-1.87	-1.50	-1.60	-0.62	-1.30	-1.41

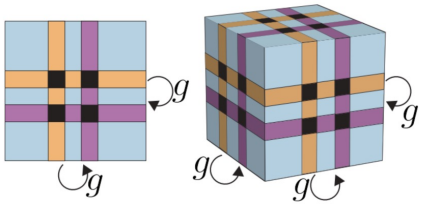


1. Single-source shortest-paths
2. Eccentricity
3. Laplacian features
4. Connected
5. Diameter
6. Spectral radius

Mean log error of different aggregators on different tasks

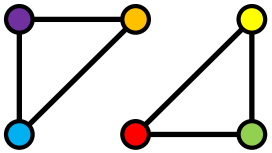
Corso et al. 2020

Towards More Expressive GNNs



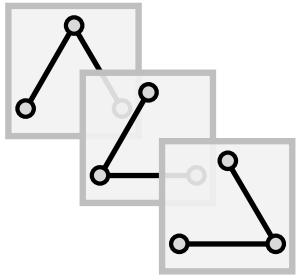
Higher-order
WL tests

Maron et al. 2019
Morris et al. 2019



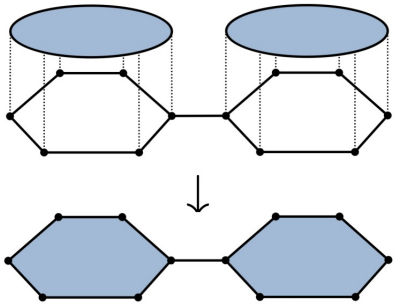
Positional &
Structural encoding

Monti, Otness et B 2018
Sato 2020
Dwivedi et al. 2020
Bouritsas, Frasca et B 2020
…many more



Subgraph
GNNs

Papp et al. 2021
Cotta et al. 2021
Zhao et al. 2021
Bevilacqua, Frasca et B, Maron 2021
Frasca et B, Maron 2022

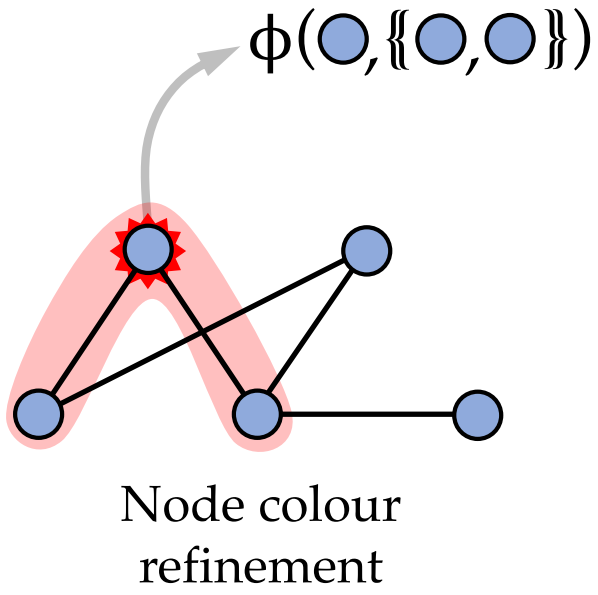


Topological
message passing

Bodnar, Frasca et B 2021

HIGHER-ORDER GNNs

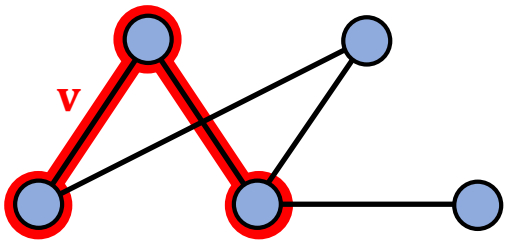
WL test



Node colour refinement

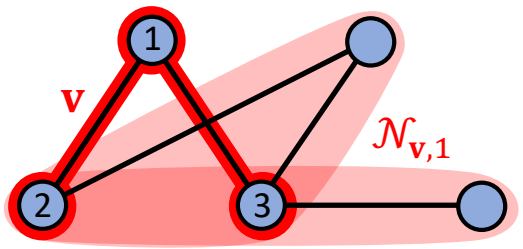
k-WL test

- Colour adjacent k-**tuples v** $\in V^k$ instead of nodes



Subgraph colour refinement

k-WL test

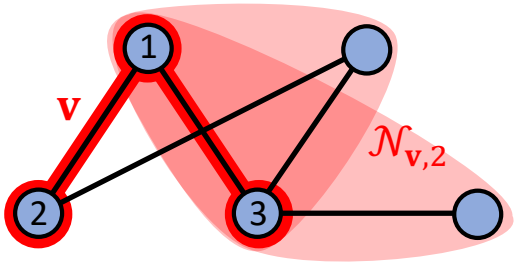


Subgraph colour refinement

- Colour adjacent k-**tuples** $\mathbf{v} \in V^k$ instead of nodes
- k-tuples are **adjacent** if they differ in one node; the j^{th} neighbourhood of $\mathbf{v}$ is defined as

$$\mathcal{N}_{\mathbf{v},j} = \{(v_1, \ldots, v_{j-1}, w, v_{j+1}, \ldots, v_k) : w \in V\}$$

k-WL test



Subgraph colour refinement

- Colour adjacent k-**tuples** $\mathbf{v} \in V^k$ instead of nodes
- k-tuples are **adjacent** if they differ in one node; the j^{th} neighbourhood of $\mathbf{v}$ is defined as

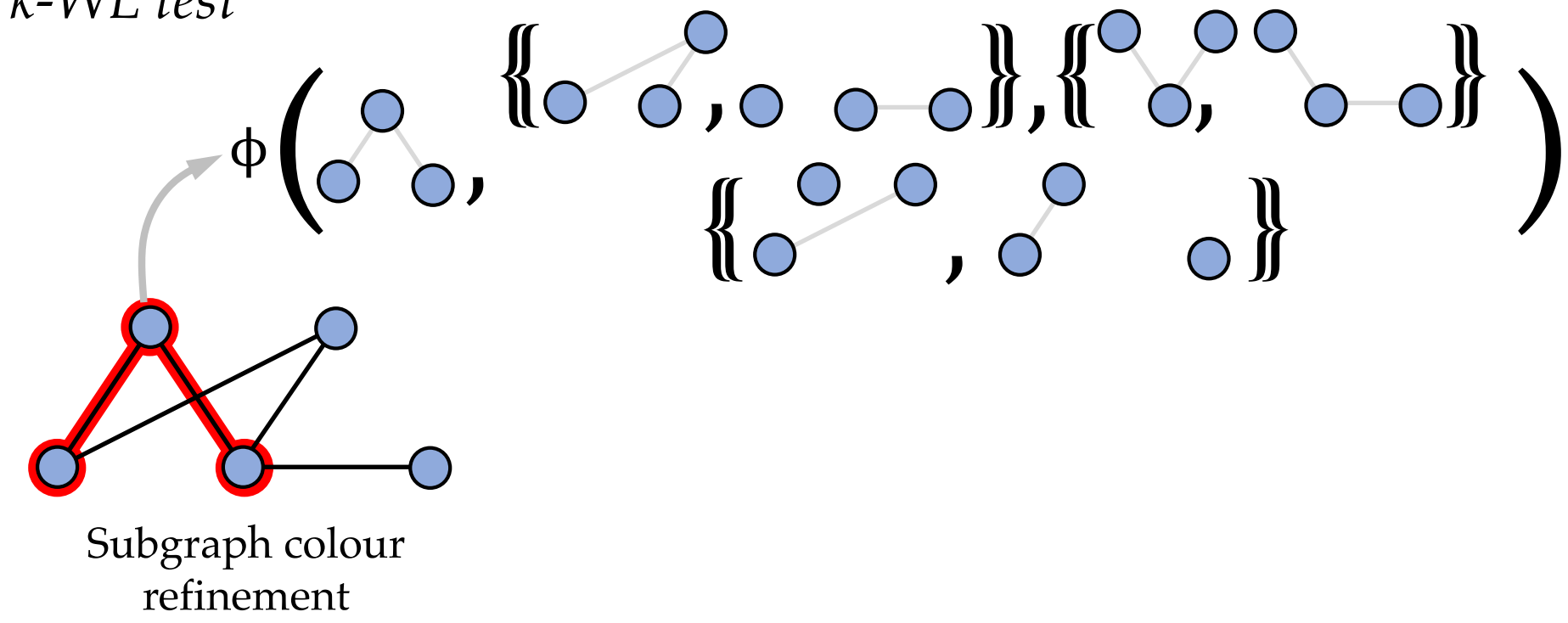
$$\mathcal{N}_{\mathbf{v},j} = \{(v_1, \dots, v_{j-1}, w, v_{j+1}, \dots, v_k) : w \in V\}$$

- For every tuple $\mathbf{v} \in V^k$, update the colour

$$\phi\left(c_{\mathbf{v}}, c_{\mathcal{N}_{\mathbf{v},1}}, \dots, c_{\mathcal{N}_{\mathbf{v},k}}\right)$$

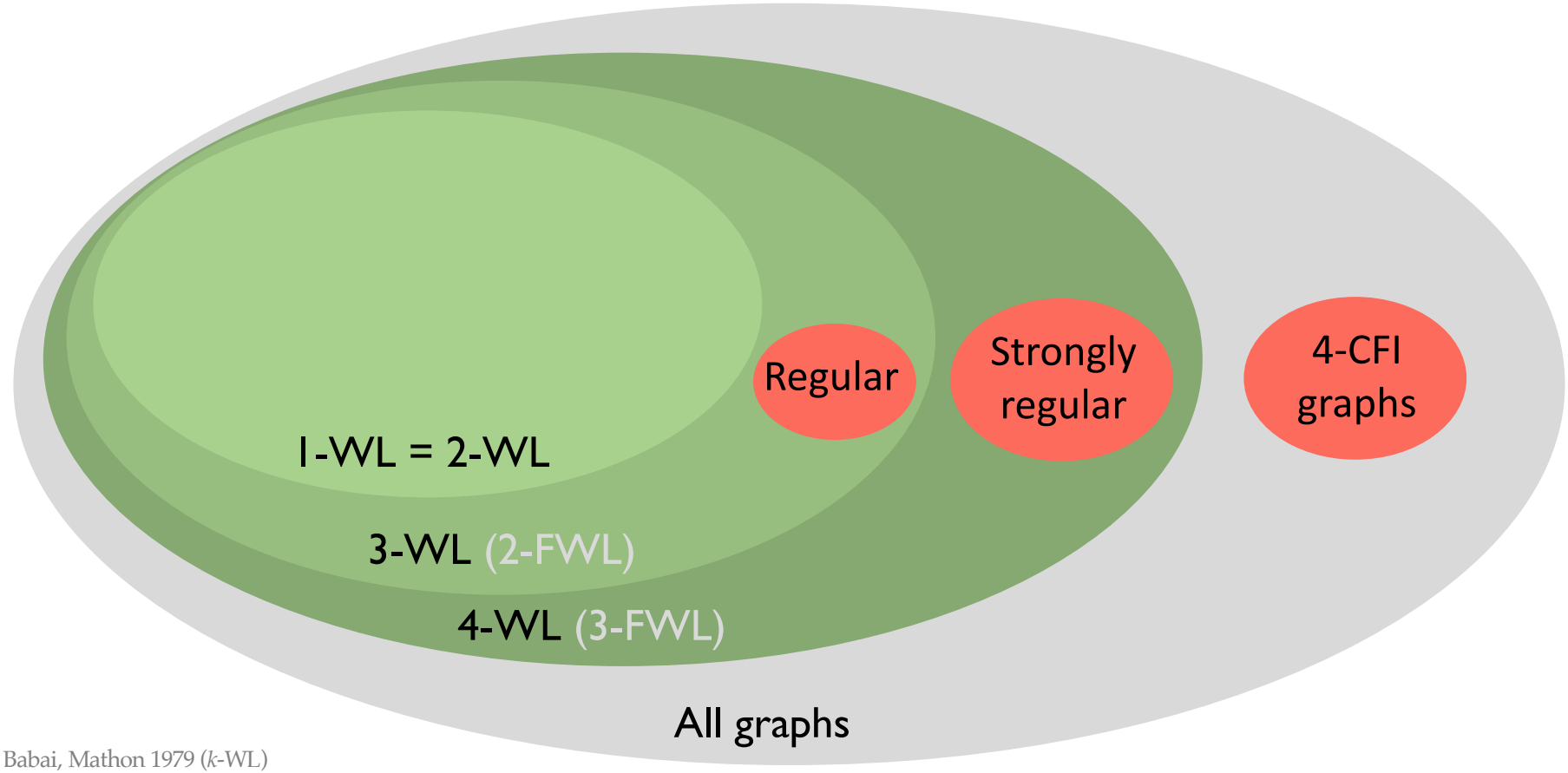
where $c_{\mathcal{N}_{\mathbf{v},j}} = \{\{c_{\mathbf{w}} : \mathbf{w} \in \mathcal{N}_{\mathbf{v},j}\}\}$

k-WL test



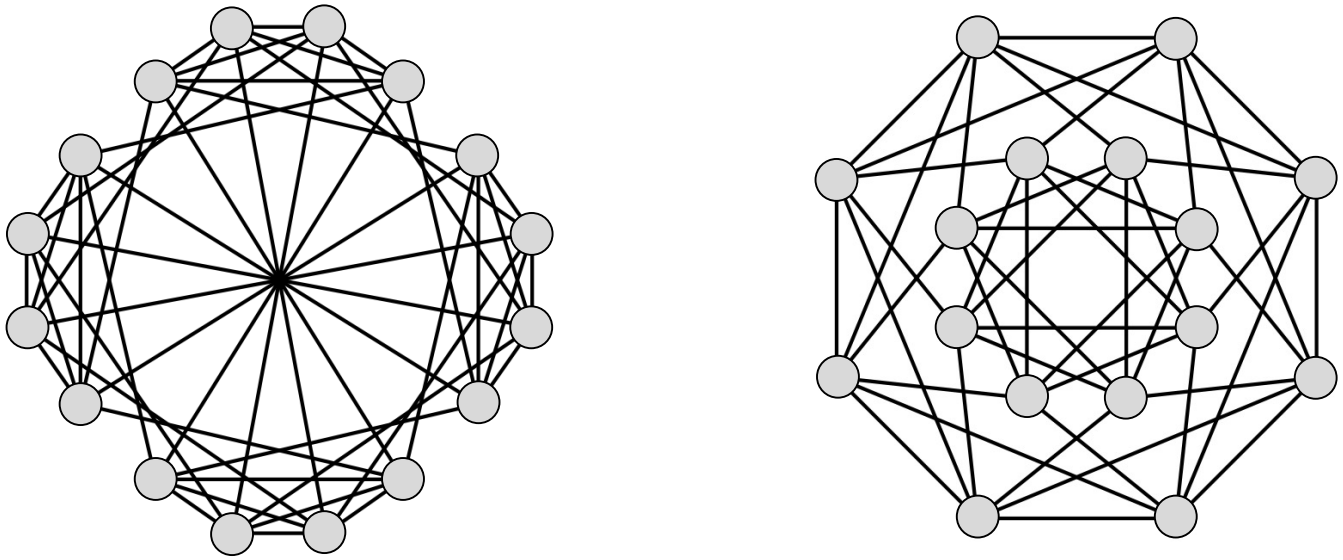
Note: there are two slightly different versions of the high-dimensional WL test referred to as "*k*-WL" and "folklore *k*-WL" that differ in the update step.

What graphs can k-WL distinguish?



Babai, Mathon 1979 (*k*-WL)
Cai, Fürer, Immerman 1992 (CFI graphs)

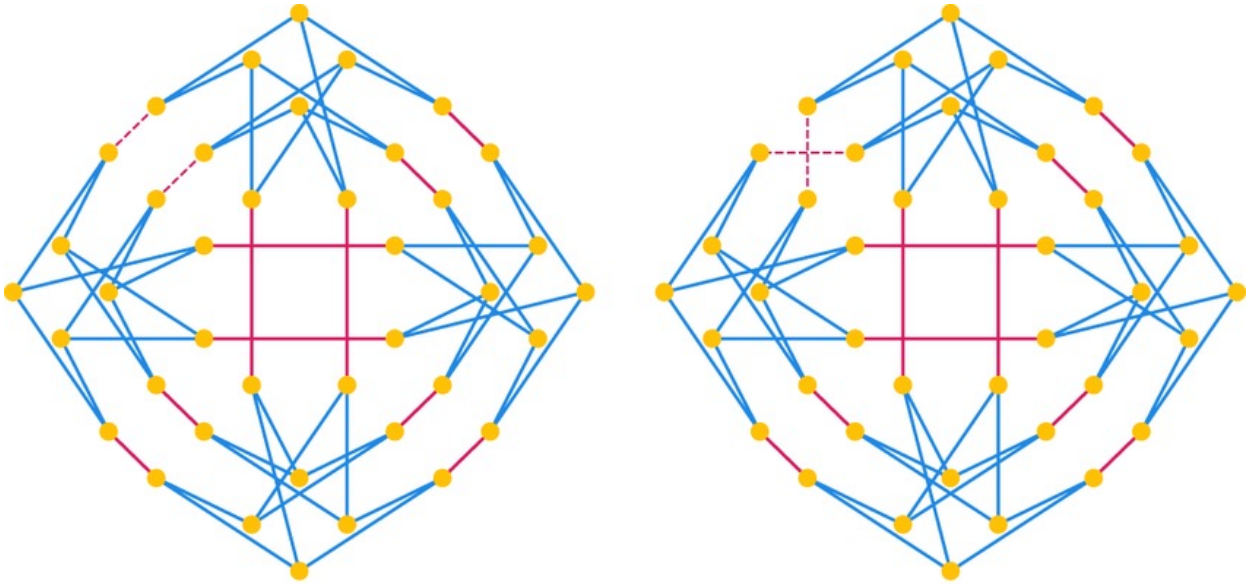
What graphs cannot 3-WL distinguish?



Strongly regular graphs cannot be distinguished by 3-WL

Bouritsas, Frasca et B 2020

What graphs cannot k-WL distinguish?



Cai-Fürer-Immerman (CFI) graphs

Cai, Fürer, Immerman 1992 (CFI graphs)

Various k-WL-like GNNs

- k-GNNs mimics the k-WL test by working with subgraphs of size k:

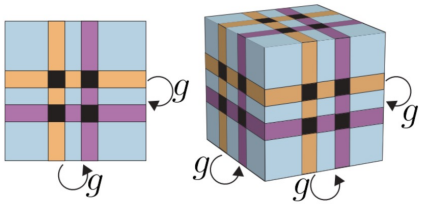
$$\mathbf{x}_{\mathbf{v}} \leftarrow \phi\left(\mathbf{W}_1 \mathbf{x}_{\mathbf{v}} + \sum_{\mathbf{u} \in \mathcal{N}_{\mathbf{v}}} \mathbf{W}_2 \mathbf{x}_{\mathbf{u}}\right)$$

  where $\mathcal{N}_{\mathbf{v}} = \{\mathbf{u} \in V^k : |\mathbf{u} \cap \mathbf{v}| = k - 1\}$

- Complexity: $\mathcal{O}(n^k)$
- More efficient variants
  - Sparse neighbourhoods
  - Sparse sets of k-tuples

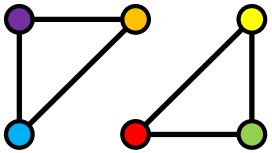
Morris et al. 2019; Morris et al. 2020; Morris et al. 2022

Towards More Expressive GNNs



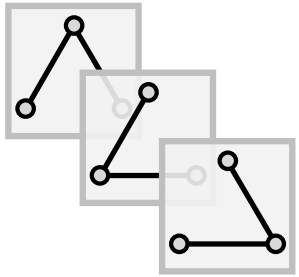
Higher-order
WL tests

Maron et al. 2019
Morris et al. 2019



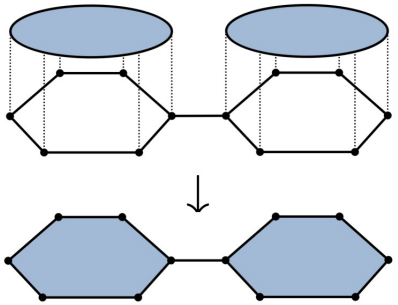
Positional &
Structural encoding

Monti, Otness et B 2018
Sato 2020
Dwivedi et al. 2020
Bouritsas, Frasca et B 2020
…many more



Subgraph
GNNs

Papp et al. 2021
Cotta et al. 2021
Zhao et al. 2021
Bevilacqua, Frasca et B, Maron 2021
Frasca et B, Maron 2022

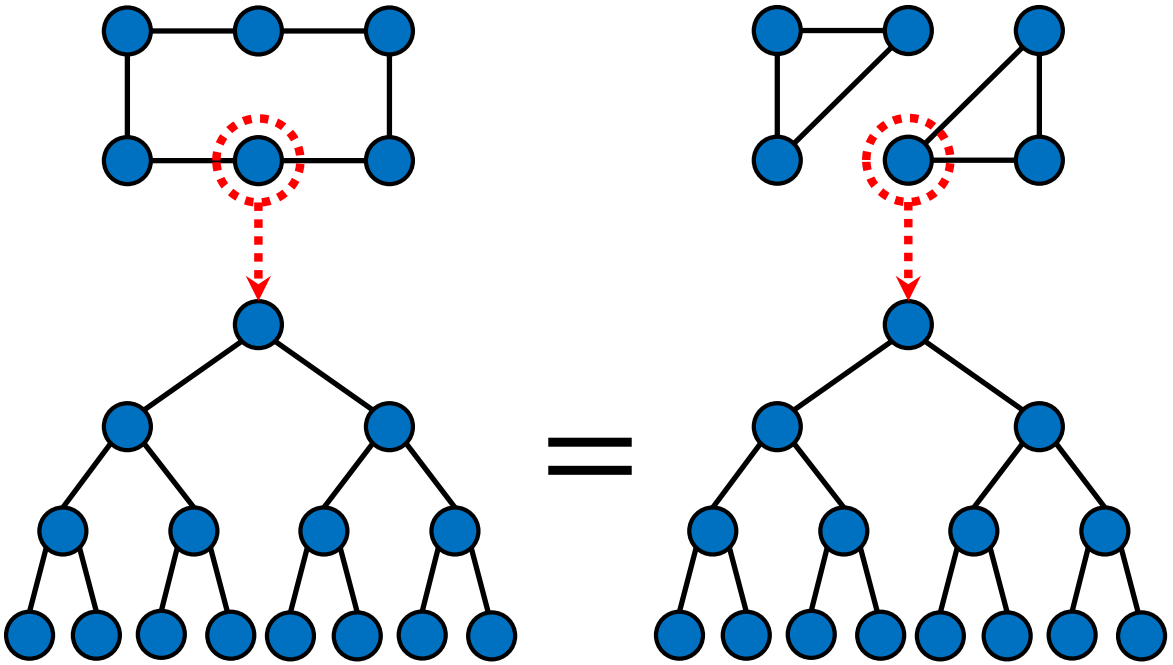


Topological
message passing

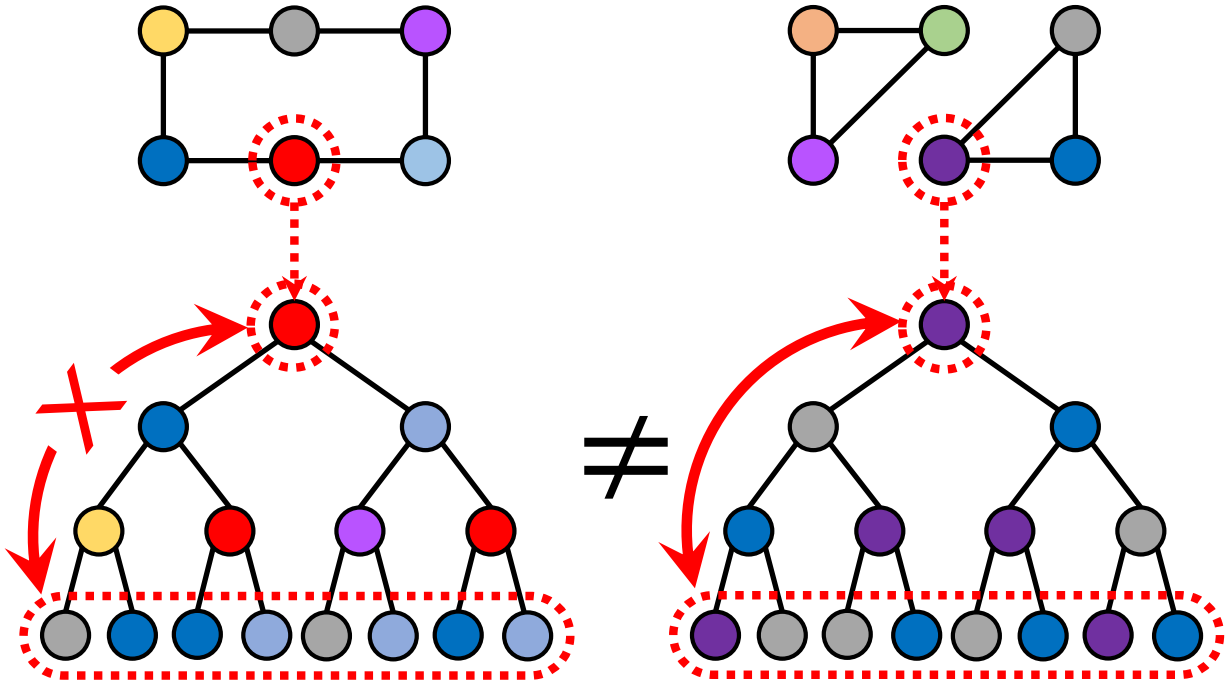
Bodnar, Frasca et B 2021

POSITIONAL ENCODING

What does WL test see?



Node encoding



“Colouring” nodes removes (some) ambiguity

How to "colour" nodes?

- Random
- Substructure count
- Laplacian/Adjacency eigenvectors
- Gradients of global encoding ("direction")
- Shortest path distance
- Diffusion kernel
- Random walk kernel
- …many more

Random node features

- rMPNN: Attach a random feature to every node of the graph, then apply MPNN
- Output of rMPNN is a *random variable*
- *Not permutation invariant!* (only in expectation)

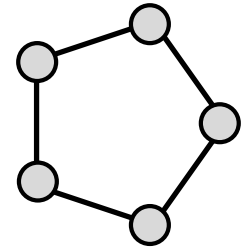
> **Probabilistic Universal Approximation:** Let f be a permutation-invariant graph function. Then, for all $\varepsilon, \delta > 0$, there exists an rMPNN $\hat{f}$ that (ε, δ)-approximates f, in the sense that
>
> $$\mathrm{P}\left(\left|f(G) - \hat{f}(G)\right| \leq \varepsilon\right) \geq 1 - \delta$$

- Embedding dimension $\mathcal{O}(n^2\delta)$
- Extensions to equivariant functions, weighted graphs, etc.

Abboud et al. 2021; Sato 2021

Graph Substructure Network (GSN)

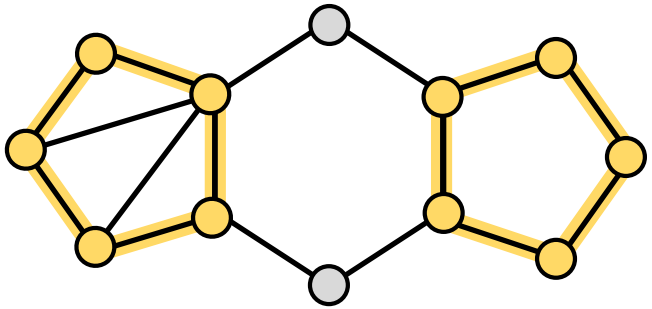
- Choose a *bank of substructures* containing graphs H of size $k = \mathcal{O}(1)$
- Count the occurrence of each H in every node/edge of the input graph
  - *Subgraph* counts
  - *Induced subgraph* counts



$H = (V_H, E_H)$

H=**subgraph** of G

$V_H \subseteq V, E_H \subseteq E$



$G = (V, E)$

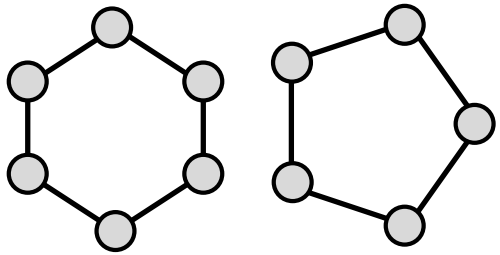
H=**induced subgraph** of G

$E_H = E \cap V_H \times V_H$

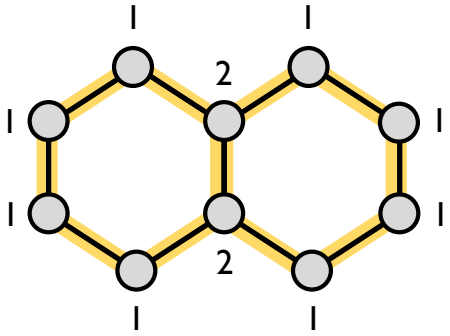
Bouritsas, Frasca et B 2020

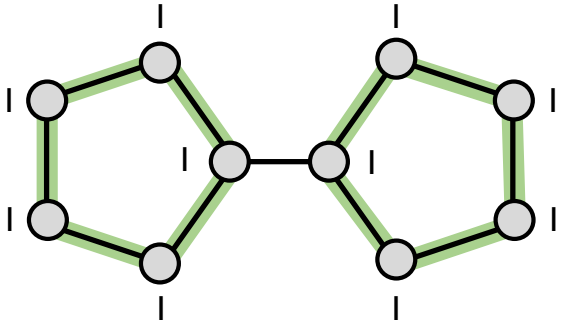
Graph Substructure Network (GSN)

- Choose a *bank of substructures* containing graphs H of size $k = \mathcal{O}(1)$
- Count the occurrence of each H in every node/edge of the input graph
- Use the counts as additional node/edge features



substructure bank

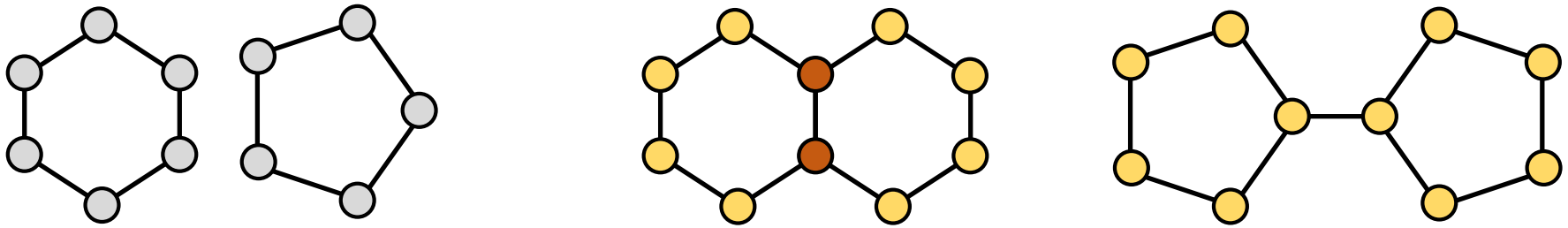




Bouritsas, Frasca et B 2020

Graph Substructure Network (GSN)

- Choose a *bank of substructures* containing graphs H of size $k = \mathcal{O}(1)$
- Count the occurrence of each H in every node/edge of the input graph
- Use the counts as additional node/edge features



substructure bank

Bouritsas, Frasca et B 2020

Graph Substructure Network (GSN)

- Choose a *bank of substructures* containing graphs H of size $k = \mathcal{O}(1)$
- Count the occurrence of each H in every node/edge of the input graph
- Use the counts as additional node/edge features
- Complexity: *precomputation* (substructure counting, which is worse case is $\mathcal{O}(n^k)$ but in practice much more optimistic) + *standard MPNN* (linear complexity $\mathcal{O}(|E|) \sim \mathcal{O}(n)$)

Theorem: GSN is strictly more expressive than WL if

- H is not a star graph, and counting is done using subgraph matching; or
- H is of size $k \geq 3$, and counting is using induced subgraph matching.

Bouritsas, Frasca et B 2020

Graph Substructure Network (GSN)

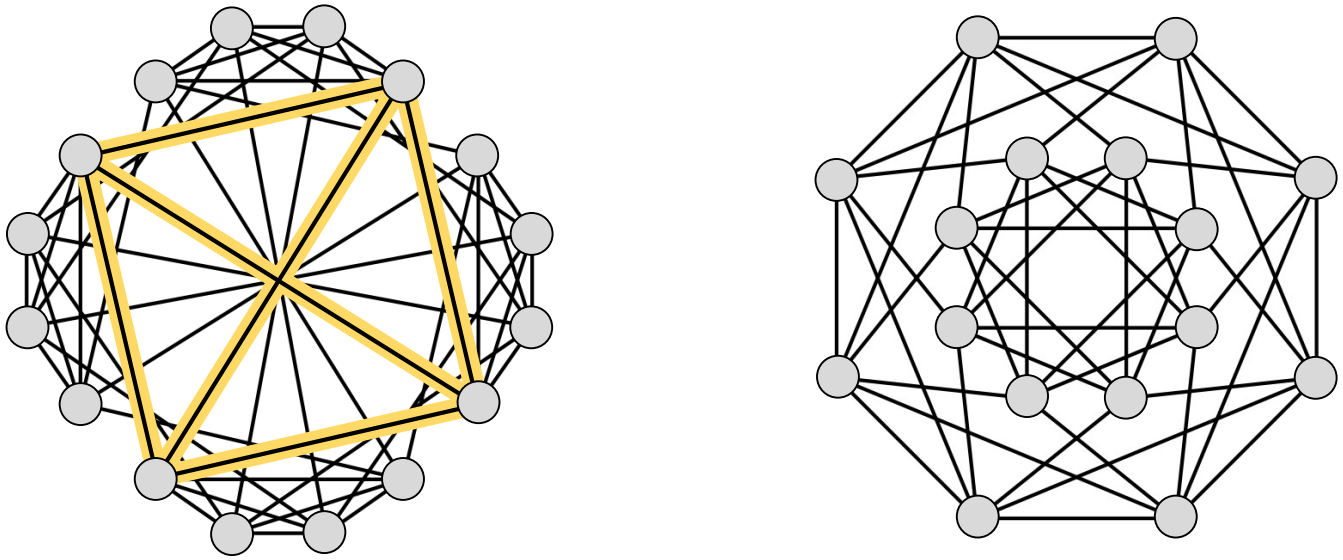
- Choose a *bank of substructures* containing graphs H of size $k = \mathcal{O}(1)$
- Count the occurrence of each H in every node/edge of the input graph
- Use the counts as additional node/edge features
- Complexity: *precomputation* (substructure counting, which is worse case is $\mathcal{O}(n^k)$ but in practice much more optimistic) + *standard MPNN* (linear complexity $\mathcal{O}(|E|) \sim \mathcal{O}(n)$)

Theorem: GSN is not less expressive than 3-WL.

Proof: by example

Bouritsas, Frasca et B 2020

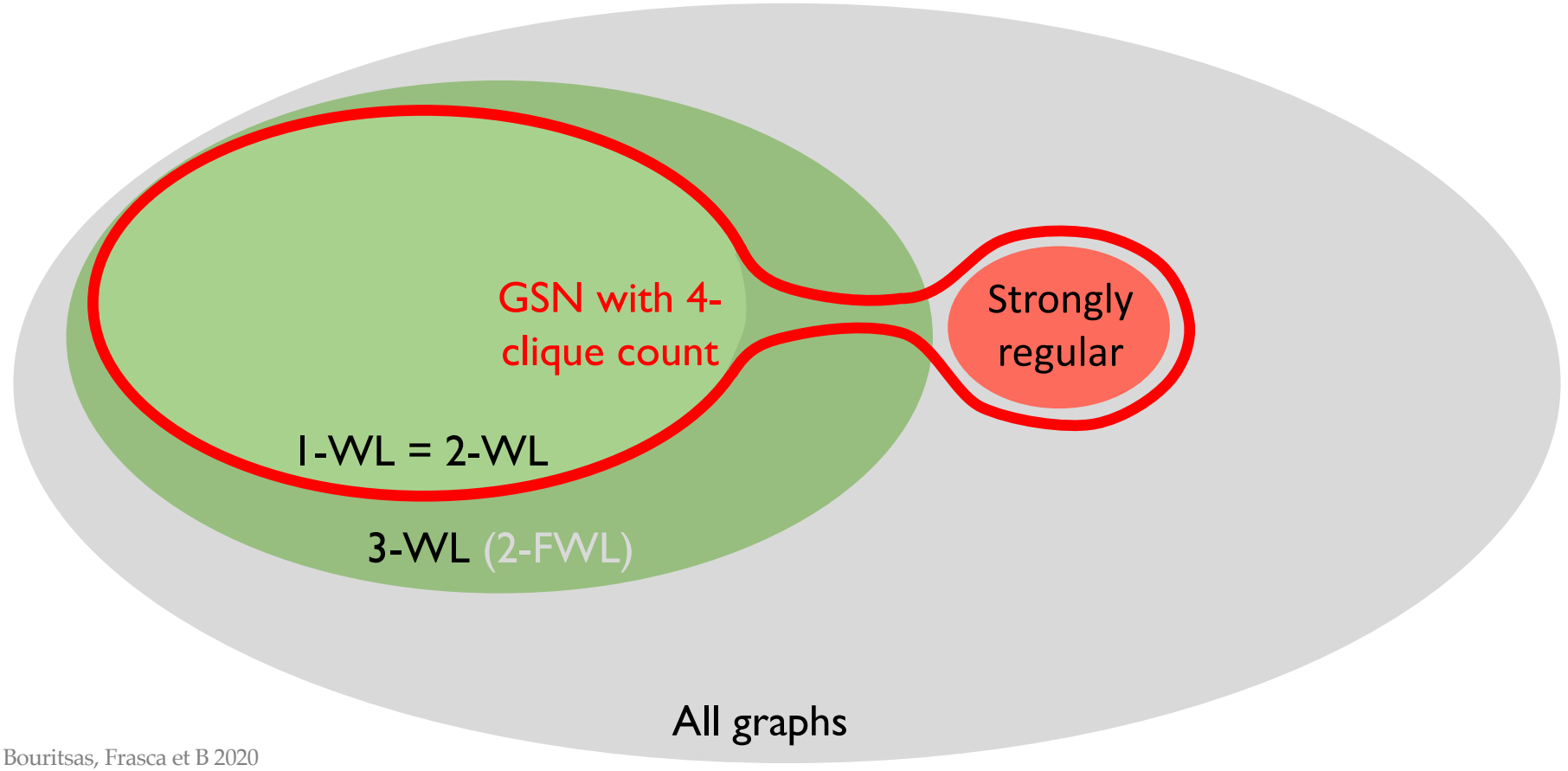
What graphs can GSN distinguish?



Strongly Regular (SR) graphs cannot be distinguished by 3-WL
but can be distinguished by GSN with 4-clique count

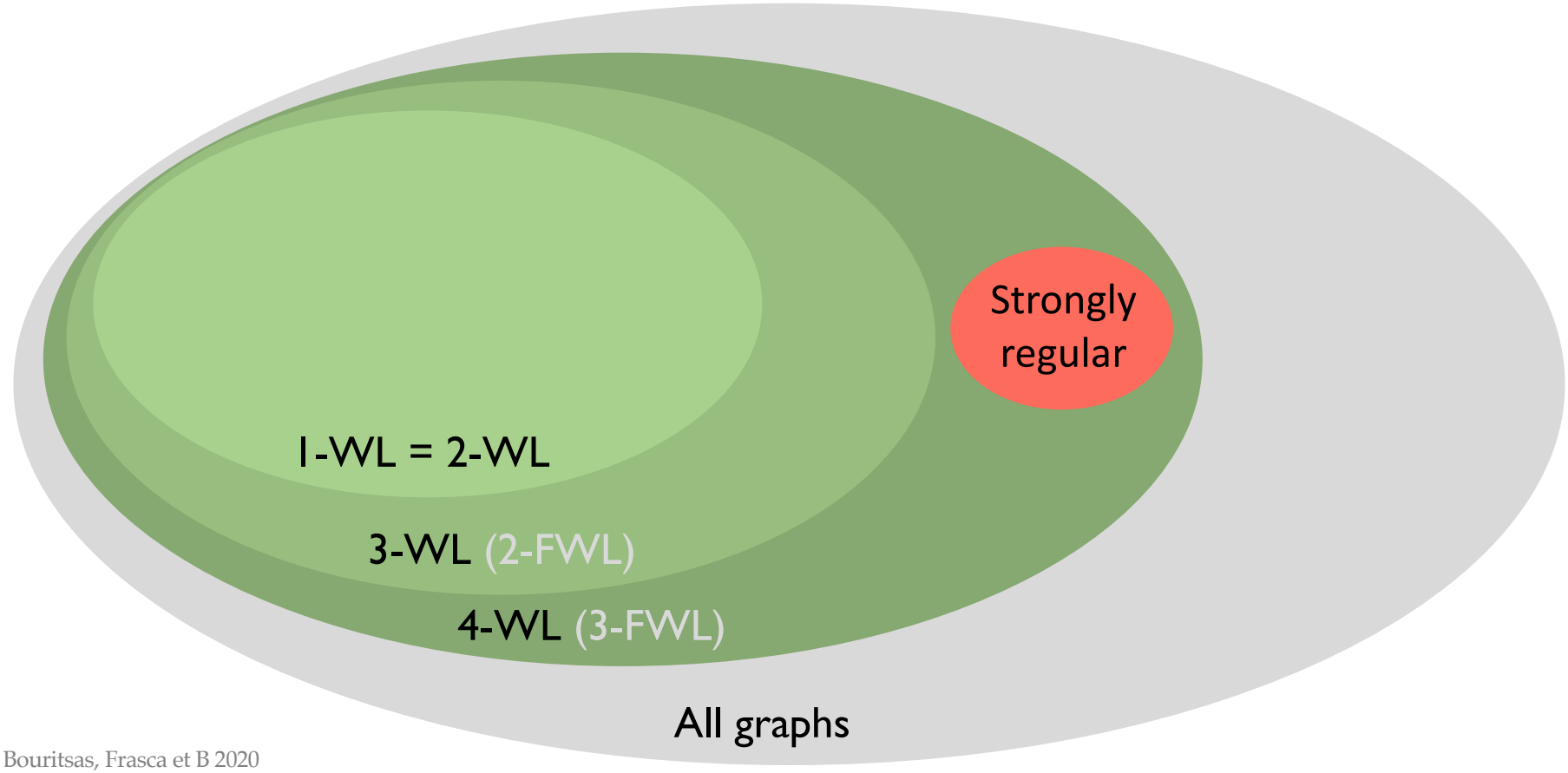
Bouritsas, Frasca et B 2020

What graphs can GSN distinguish?



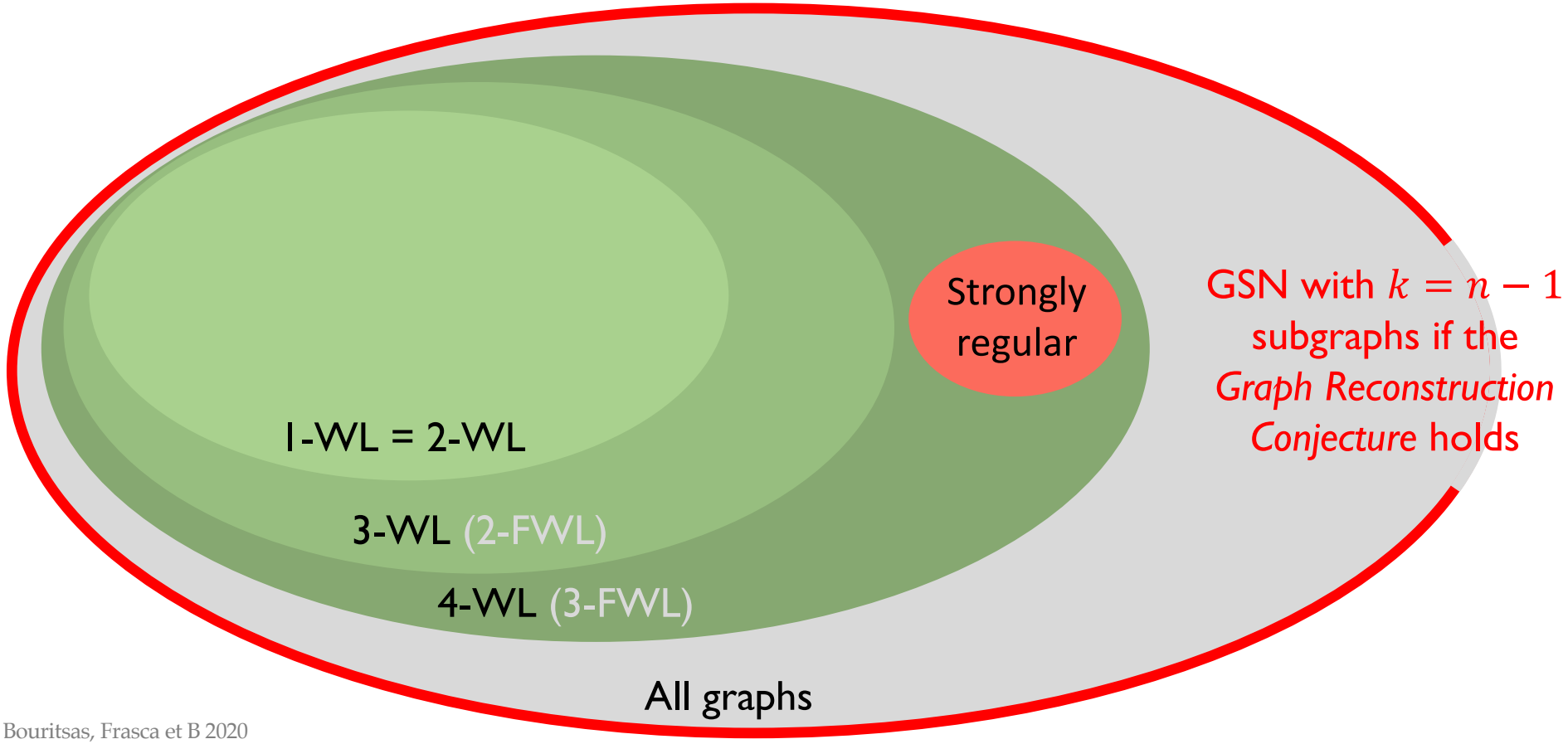
Bouritsas, Frasca et B 2020

What graphs can GSN distinguish hypothetically?



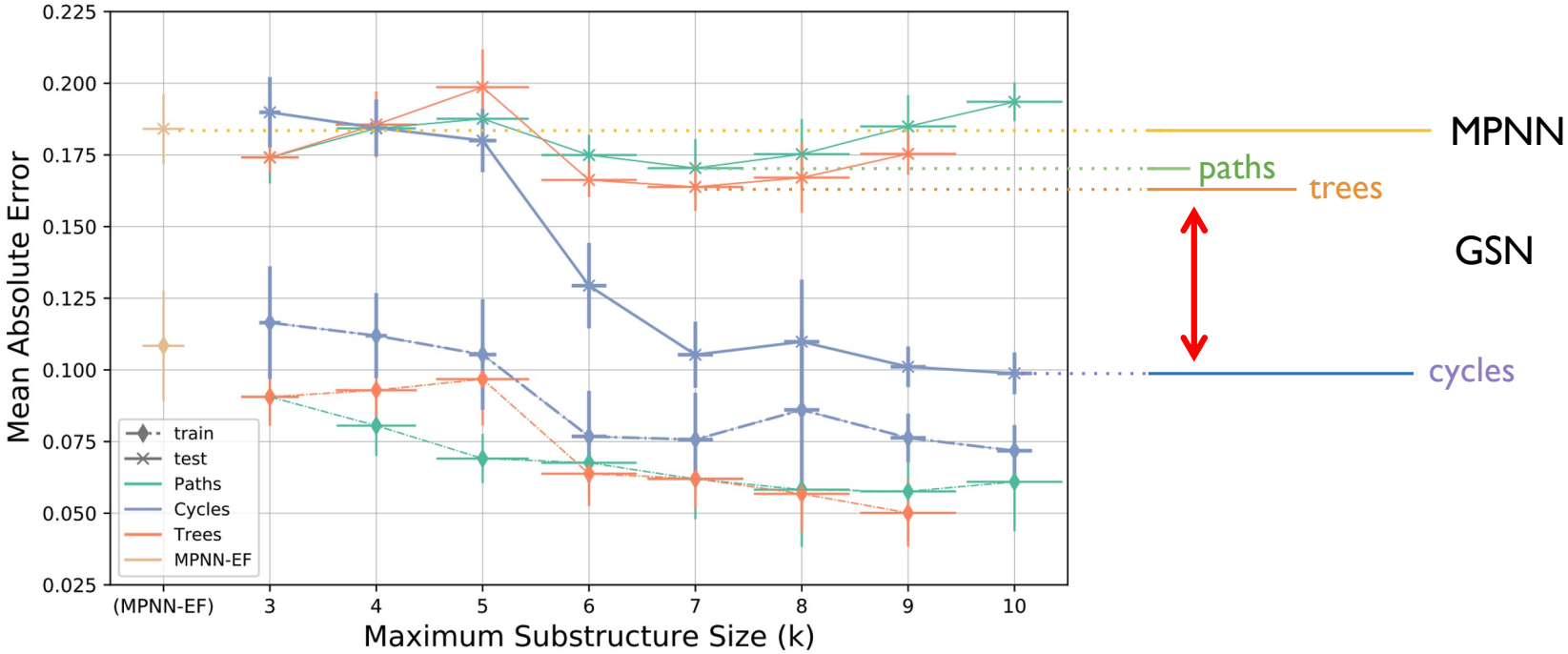
Bouritsas, Frasca et B 2020

What graphs can GSN distinguish hypothetically?



Bouritsas, Frasca et B 2020

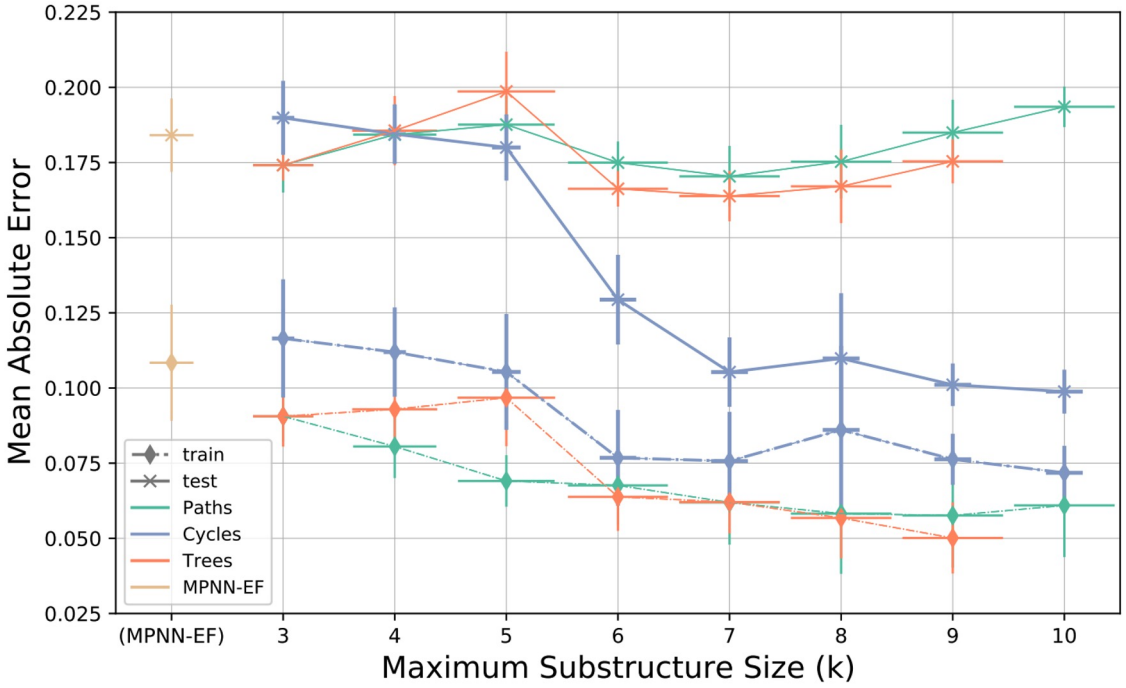
Graph Substructure Network in practice



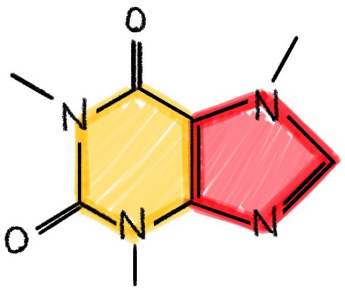
Molecule property prediction on ZINC
using GSN with different substructures

Bouritsas, Frasca et B 2020
Slide adapted from Bevilacqua et al. (LoG 2022 tutorial)

Graph Substructure Network in practice



Molecule property prediction on ZINC
using GSN with different substructures

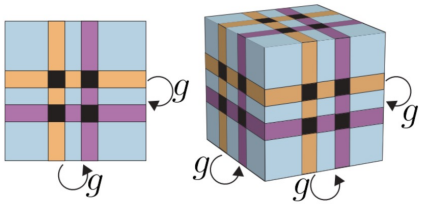


Molecule of caffeine

Substructure = application-specific inductive bias

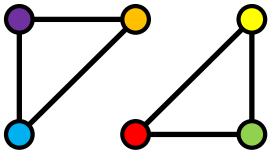
Bouritsas, Frasca et B 2020
Slide adapted from Bevilacqua et al. (LoG 2022 tutorial)

Towards More Expressive GNNs



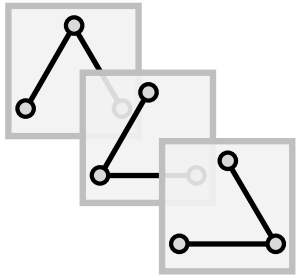
Higher-order
WL tests

Maron et al. 2019
Morris et al. 2019



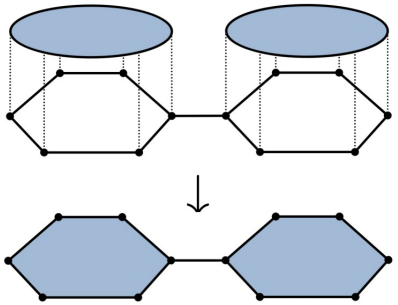
Positional &
Structural encoding

Monti, Otness et B 2018
Sato 2020
Dwivedi et al. 2020
Bouritsas, Frasca et B 2020
…many more



Subgraph
GNNs

Papp et al. 2021
Cotta et al. 2021
Zhao et al. 2021
Bevilacqua, Frasca et B, Maron 2021
Frasca et B, Maron 2022

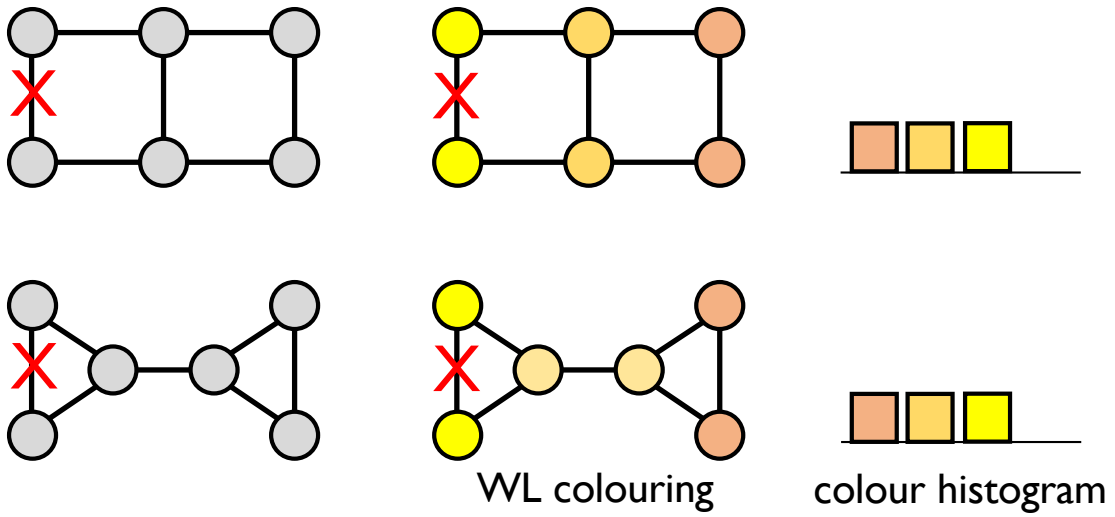


Topological
message passing

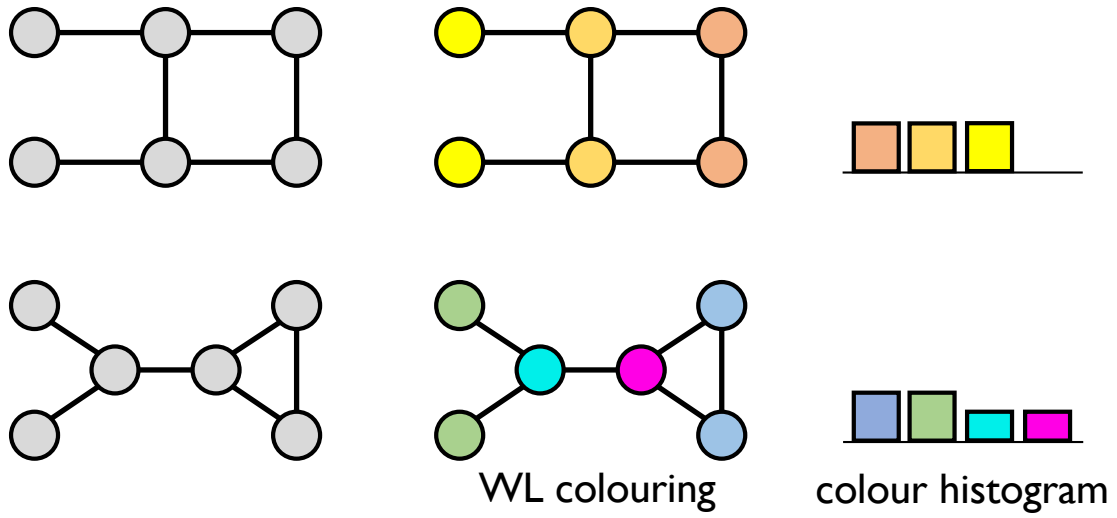
Bodnar, Frasca et B 2021

SUBGRAPH GNNs

Subgraph GNNs



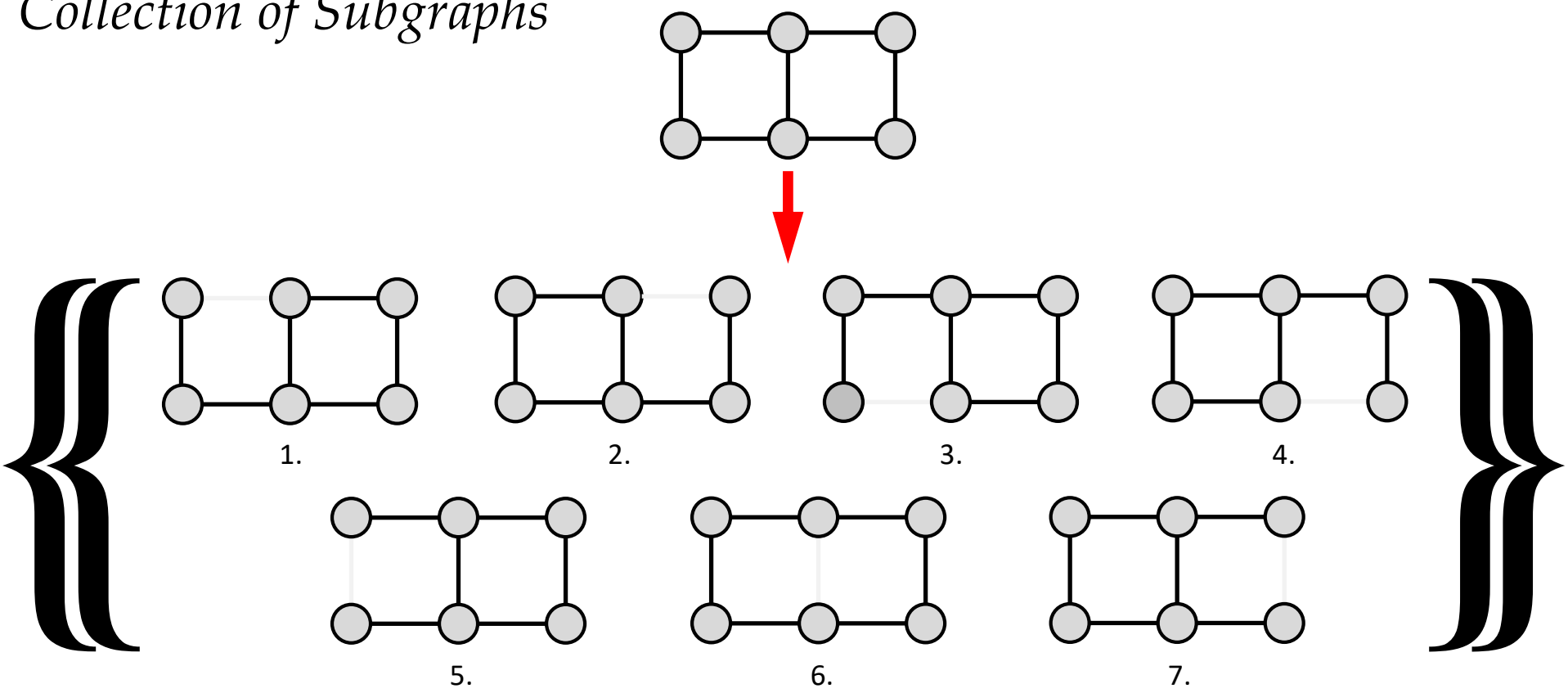
Subgraph GNNs



Graph perturbation allows to distinguish between structures otherwise indistinguishable by Weisfeiler-Lehman

Papp et al. 2021; Cotta et al. 2021; Zhao et al. 2021 Bevilacqua, Frasca et B, Maron 2022; Frasca et B, Maron 2022

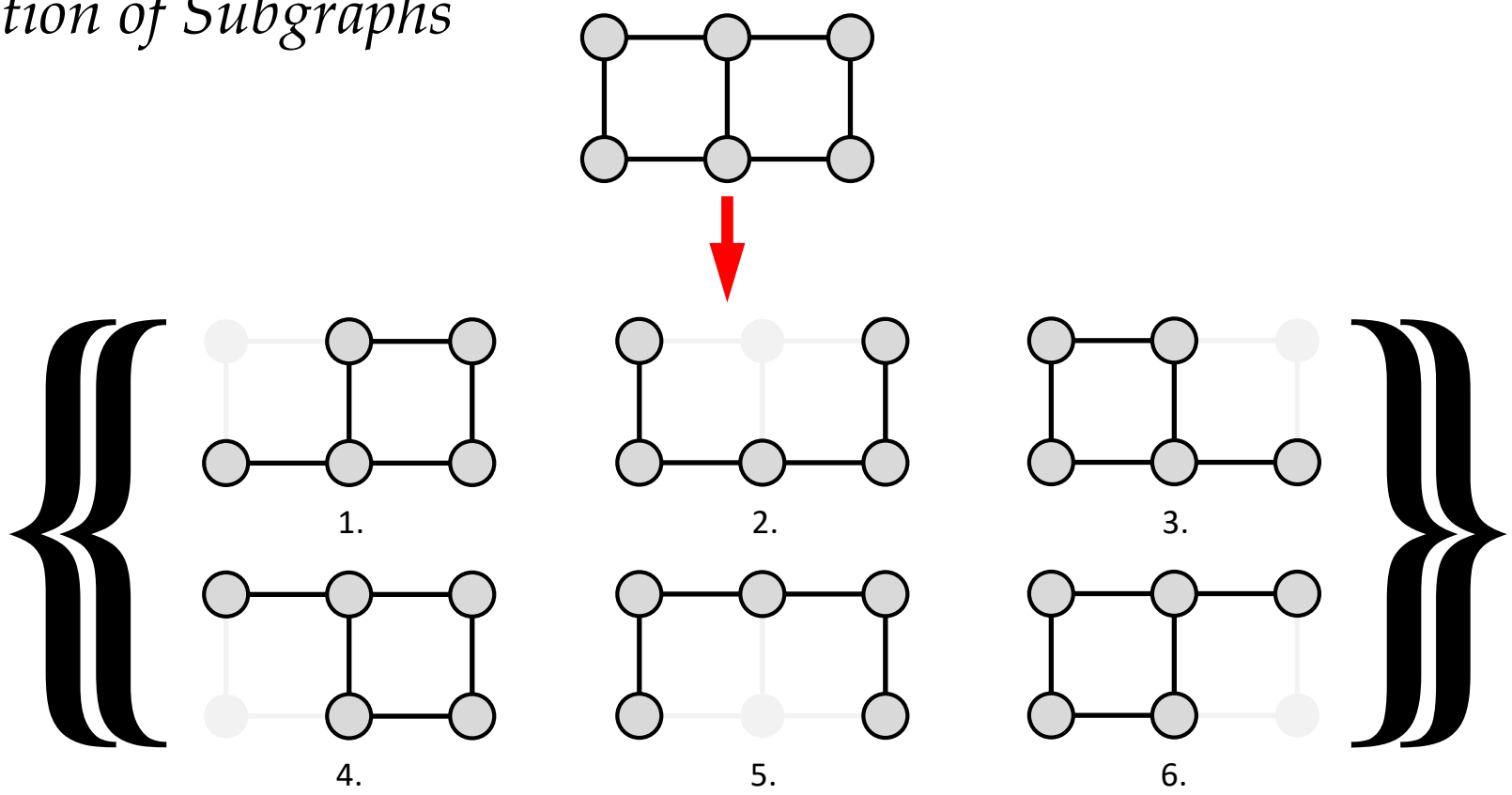
Collection of Subgraphs



A multiset of subgraphs obtained by edge deletion

Bevilacqua, Frasca et B, Maron 2022; Frasca et B, Maron 2022

Collection of Subgraphs



A multiset of subgraphs obtained by node deletion

Bevilacqua, Frasca et B, Maron 2022; Frasca et B, Maron 2022

Graph Reconstruction Conjectures

Graph Reconstruction Conjecture: A graph H is said to be a *reconstruction* of G (denoted $H \sim G$) if they have the same multiset (*deck*) of node-removed subgraphs (*cards*). If G and H are two finite, undirected, simple graphs with at least three vertices and H is the reconstruction of G, then $H \simeq G$.

P. Kelly

S. Ulam

Collection of subgraphs determines the graph isomorphism class

Kelly 1942 (PhD thesis where the Conjecture appeared); Kelly 1957 (reconstruction from *k*-subgraphs); McKay 1997 (proof for small graphs); Nýdl 2001 (k-reconstructability for certain graph families)

Graph Reconstruction Conjectures

Graph Reconstruction Conjecture: A graph H is said to be a *reconstruction* of G (denoted $H \sim G$) if they have the same multiset (*deck*) of node-removed subgraphs (*cards*). If G and H are two finite, undirected, simple graphs with at least three vertices and H is the reconstruction of G, then $H \simeq G$.

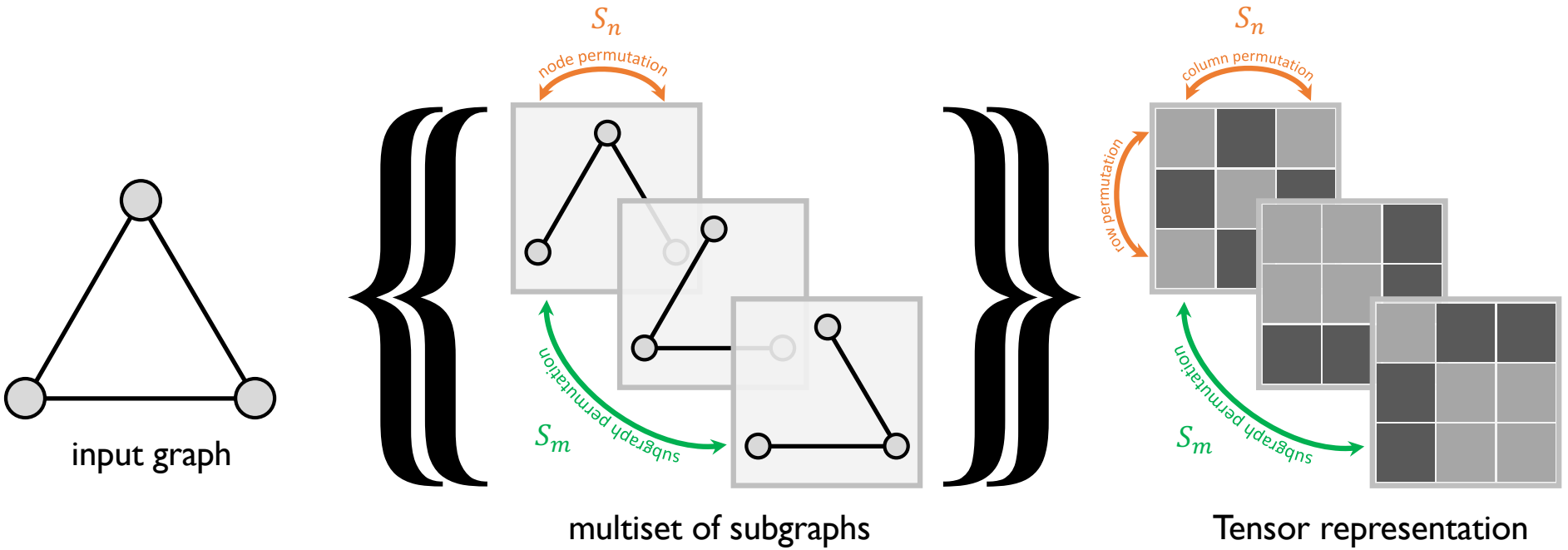
P. Kelly

S. Ulam

- Proven for small graphs with $n \leq 11$ nodes
- Open question in general
- Generalisations for subgraphs of size $n - k$

Kelly 1942 (PhD thesis where the Conjecture appeared); McKay 1997 (proof for small graphs);
Kelly 1957 (reconstruction from *k*-subgraphs); Nýdl 2001 (k-reconstructability for certain graph families)

Equivariant Subgraph Aggregation Networks

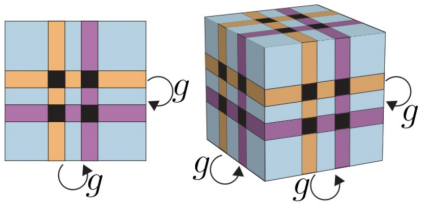


Symmetry group of subgraph colection $G = S_n \times S_m$

Bevilacqua, Frasca et B, Maron 2022; Frasca et B, Maron 2022

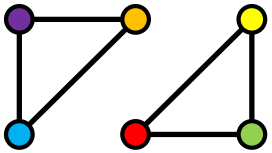

G_4
G_3
G_3
G_1
B et al. 2022

Towards More Expressive GNNs



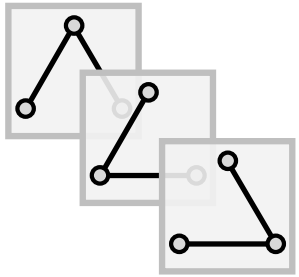
Higher-order
WL tests

Maron et al. 2019
Morris et al. 2019



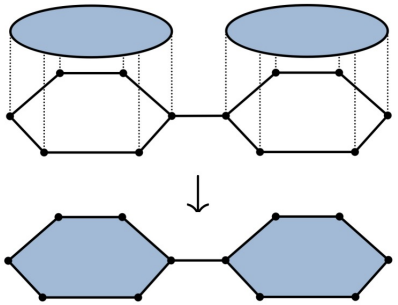
Positional &
Structural encoding

Monti, Otness et B 2018
Sato 2020
Dwivedi et al. 2020
Bouritsas, Frasca et B 2020
…many more



Subgraph
GNNs

Papp et al. 2021
Cotta et al. 2021
Zhao et al. 2021
Bevilacqua, Frasca et B, Maron 2021
Frasca et B, Maron 2022

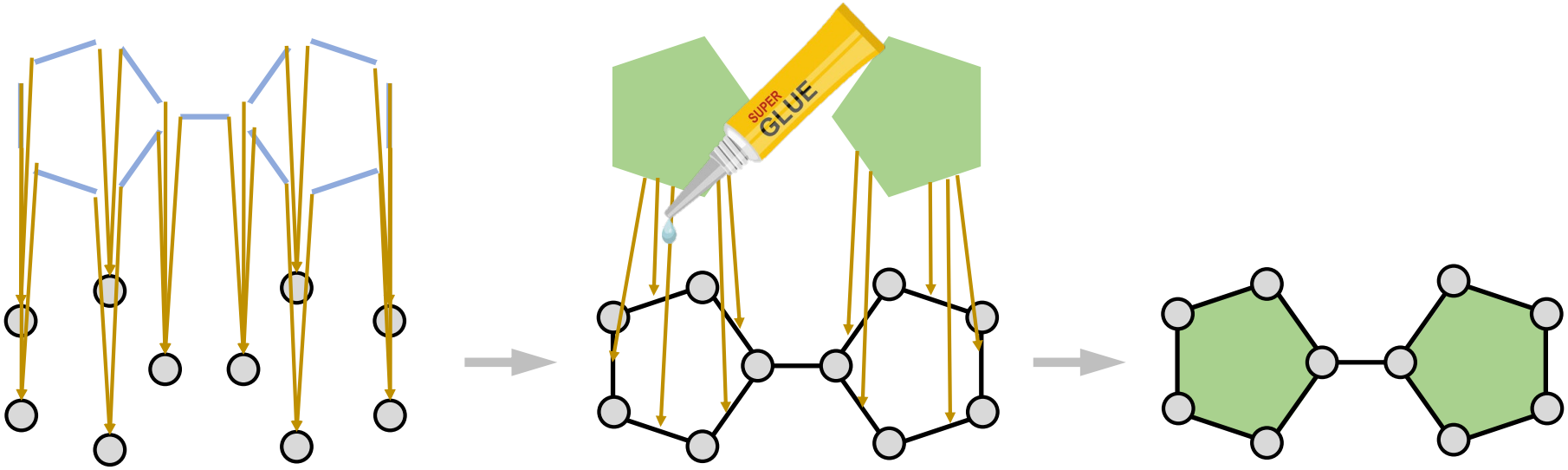


Topological
message passing

Bodnar, Frasca et B 2021

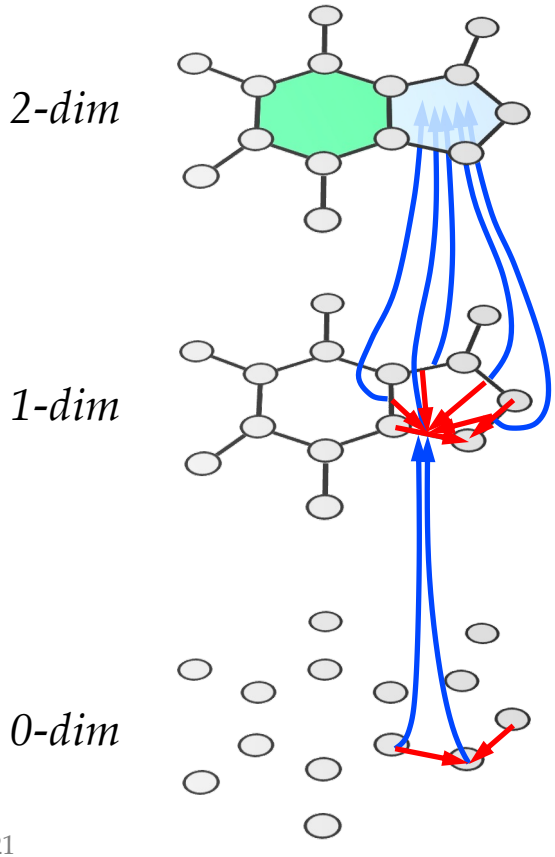
TOPOLOGICAL MESSAGE PASSING

Cell complexes



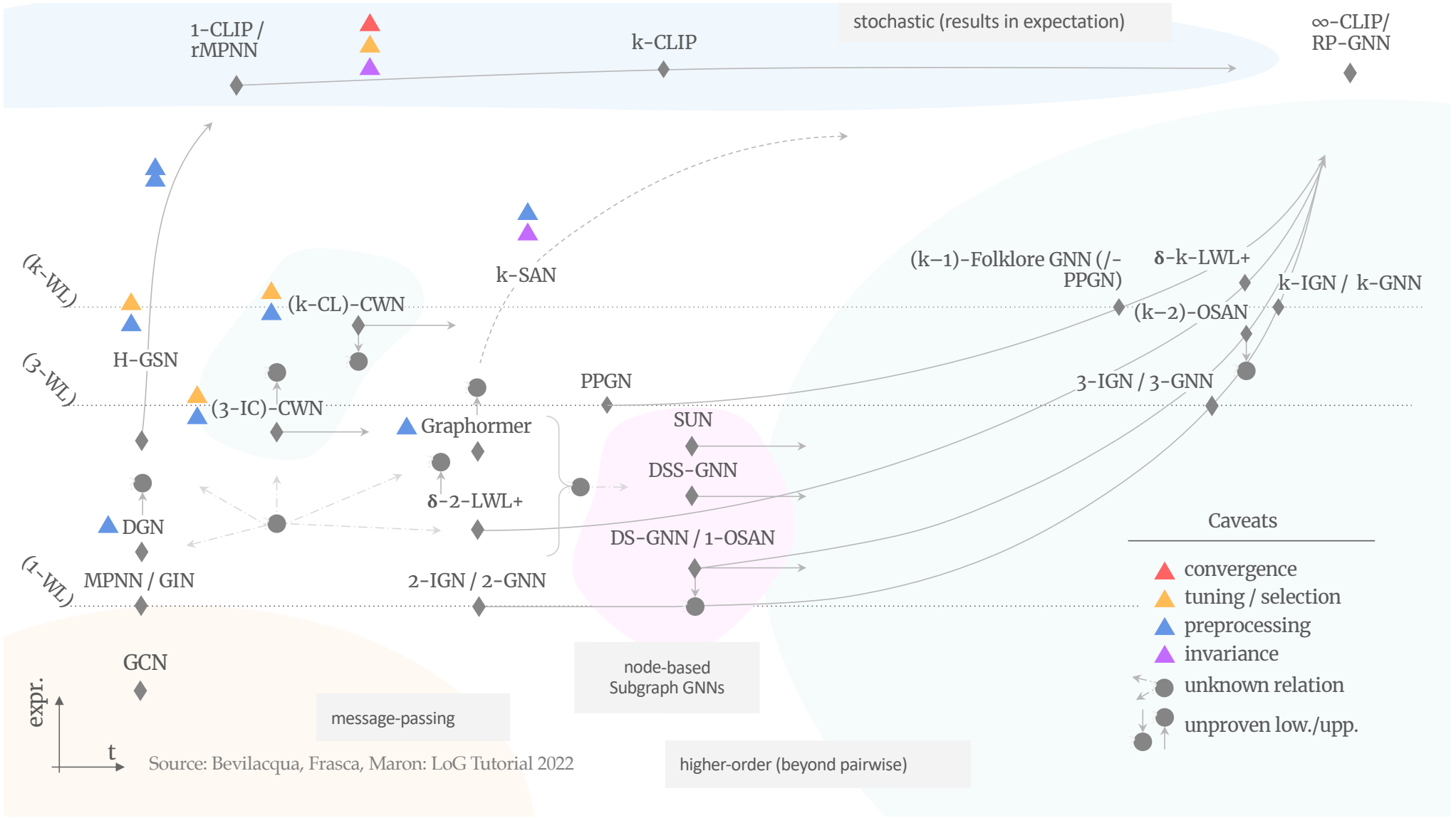
Set **Graph** **Cell (CW) complex**

Topological Message Passing



- Lift the graph into a cell complex
- Hierarchical message passing
- Strictly more expressive than Weisfeiler-Lehman

Bodnar, Frasca et B. 2021


stochastic (results in expectation)
1-CLIP / rMPNN
k-CLIP
∞-CLIP/ RP-GNN
(k-WL)
(3-WL)
(1-WL)
H-GSN
(k-CL)-CWN
k-SAN
(k−1)-Folklore GNN (/- PPGN)
δ-k-LWL+
k-IGN / k-GNN
(k−2)-OSAN
(3-IC)-CWN
PPGN
3-IGN / 3-GNN
Graphormer
SUN
DSS-GNN
δ-2-LWL+
DGN
DS-GNN / 1-OSAN
MPNN / GIN
2-IGN / 2-GNN
Caveats
convergence
tuning / selection
preprocessing
invariance
unknown relation
unproven low./upp.
GCN
expr.
t
message-passing
node-based Subgraph GNNs
higher-order (beyond pairwise)
Source: Bevilacqua, Frasca, Maron: LoG Tutorial 2022

Takeaways

- Message-Passing GNNs are upper-bounded in their expressive power by the Weisfeiler-Lehman graph isomorphism test
- Several ways to enhance expressive power:
  - Higher-order WL-tests
  - Structural and positional encoding
  - Subgraph aggregation methods
  - Graph lifting to simplicial/cellular complexes & topological message passing
- Generalisation power of GNNs is an open question
- Next: is Weisfeiler-Lehman formalism the end of story? (No!)

Key Concepts

- Graph isomorphism
- Weisfeiler-Lehman tests
- Positional and structural coding
- Reconstruction Conjectures

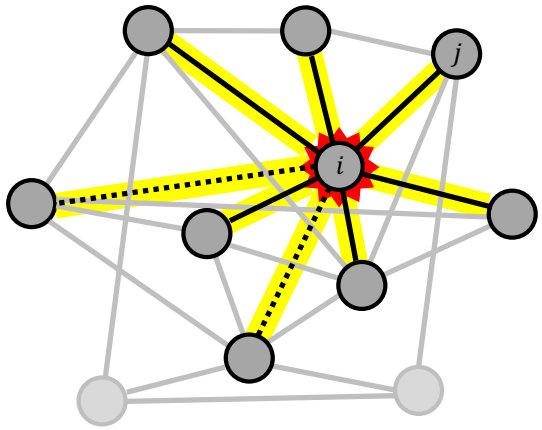
Main References

- M. Bronstein et al., Geometric deep learning, *arXiv:2104.13478, 2021*. Section 4.1 "Graphs and sets" and Section 5.3 "Graph neural networks"
- C. Morris et al., Weisfeiler and Leman go Machine Learning: The Story so far, *arXiv:2112.099992*, 2021. Introductory text on *k*-WL tests and various equivalent GNNs
- B. Bevilacqua, F. Frasca, H. Maron, Exploring the practical and theoretical landscape of expressive GNNs, LoG tutorial 2022. Expressive power and advanced GNN architectures

Additional References

- Z. Chen et al., On the equivalence between graph isomorphism testing and function approximation with GNNs, *NeurIPS* 2019. Universal approximation in GNNs
- K. Xu et al., How powerful are graph neural networks?, *ICLR* 2019. Proof of equivalence between WL-test and MPNN + GIN architecture
- M. Bronstein, L. Cotta, F. Frasca, H. Maron, Using subgraphs for more expressive GNNs, Towards Data Science blog post 2021. Diverse subgraph methods + Reconstruction conjectures
- C. Bodnar, F. Frasca, et al., Weisfeiler and Lehman go cellular: CW networks, *NeurIPS* 2021. Topological message passing

Main Ingredients of an MPNN



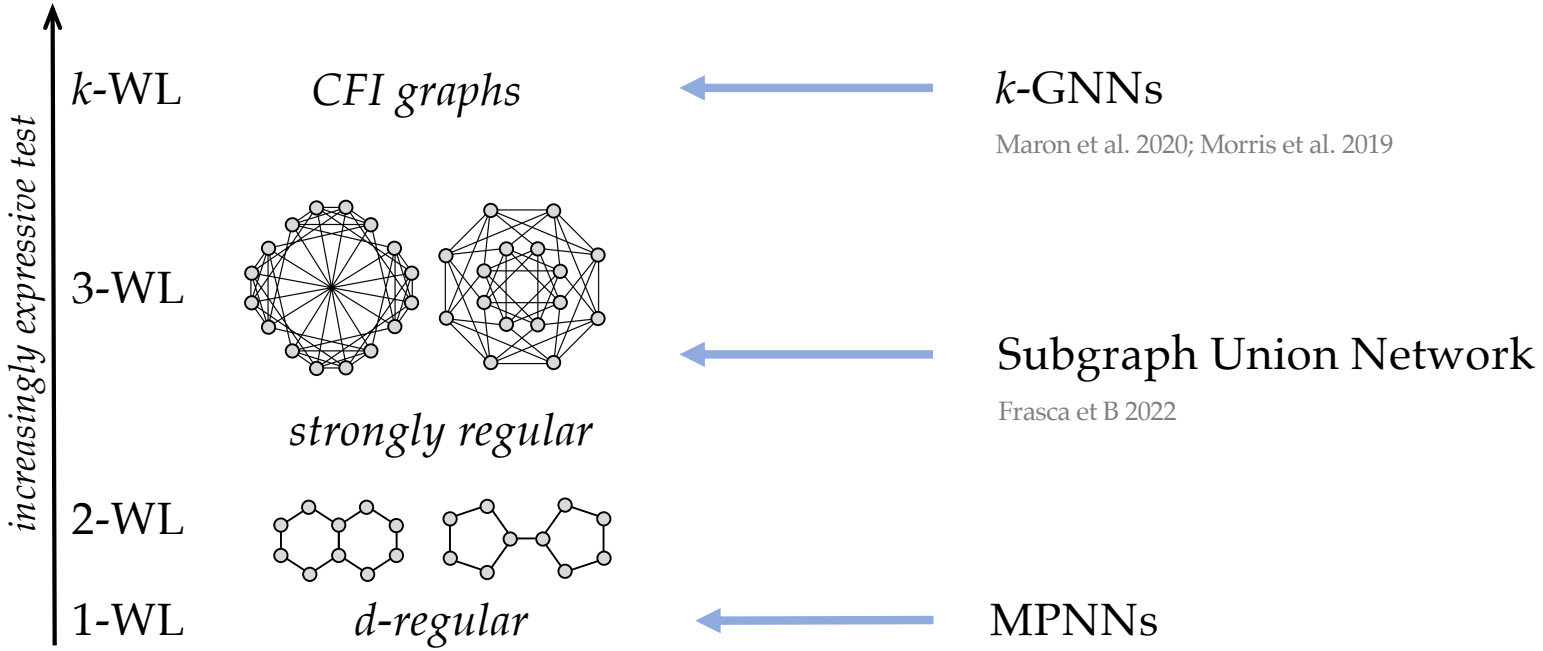
$$\mathbf{x}_i \leftarrow \phi\left(\mathbf{x}_i, \square_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

Message passing function

Aggregation operator

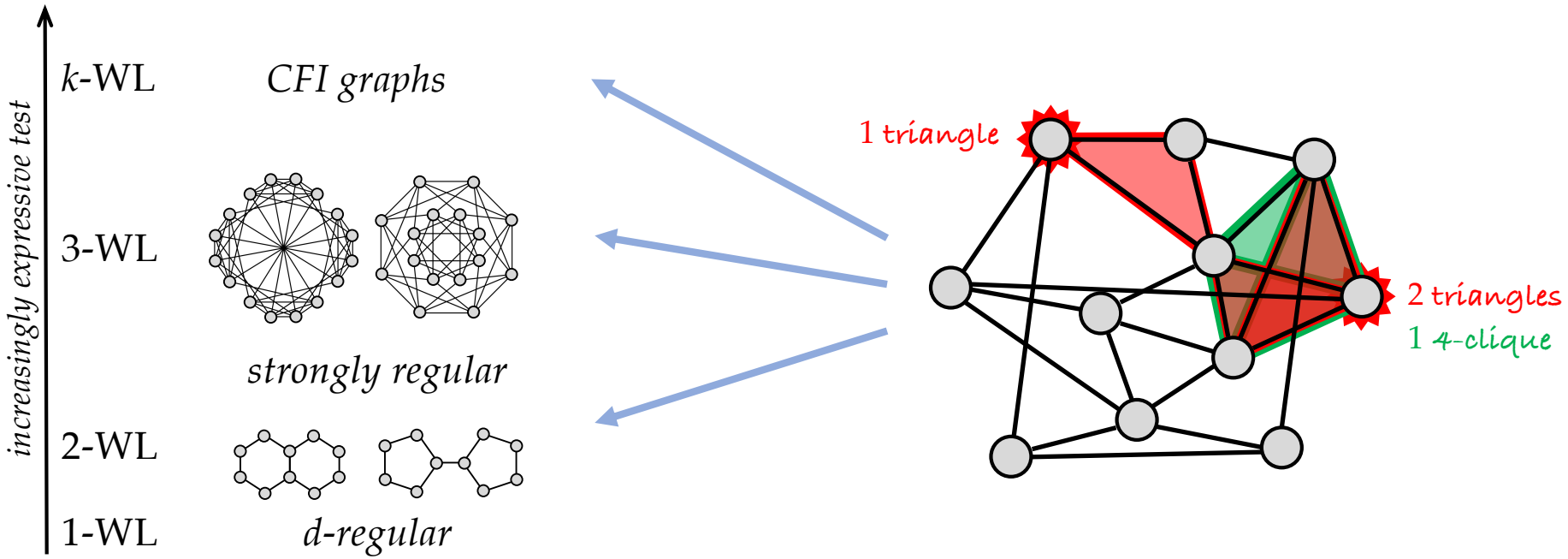
Computational graph

Weisfeiler-Lehman hierarchy



Weisfeiler, Lehman 1968 (2-WL); Babai, Mathon 1979 (*k*-WL)
Cai, Furer, Immerman 1992 (CFI graphs)

Weisfeiler-Lehman hierarchy



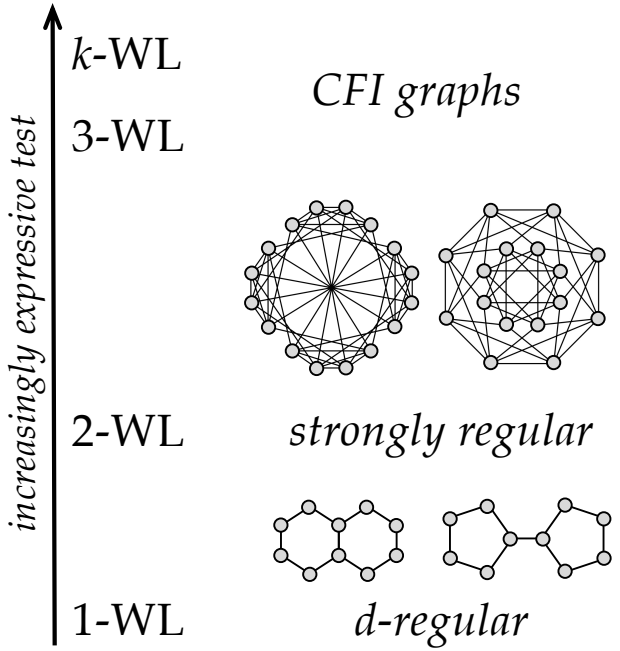
GNN expressive power

Structural encoding

Weisfeiler, Lehman 1968 (2-WL); Babai, Mathon 1979 (*k*-WL);
Cai, Furer, Immerman 1992 (CFI graphs)

Bouritsas, Frasca et B 2020

Weisfeiler-Lehman hierarchy

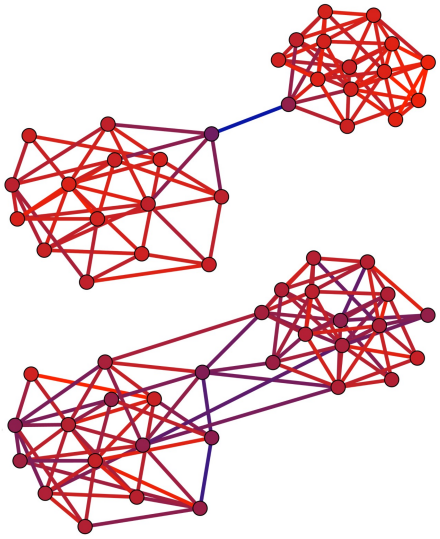


GNN expressive power

Weisfeiler, Lehman 1968 (2-WL); Babai, Mathon 1979 (*k*-WL); Cai, Fürer, Immerman 1992 (CFI graphs)

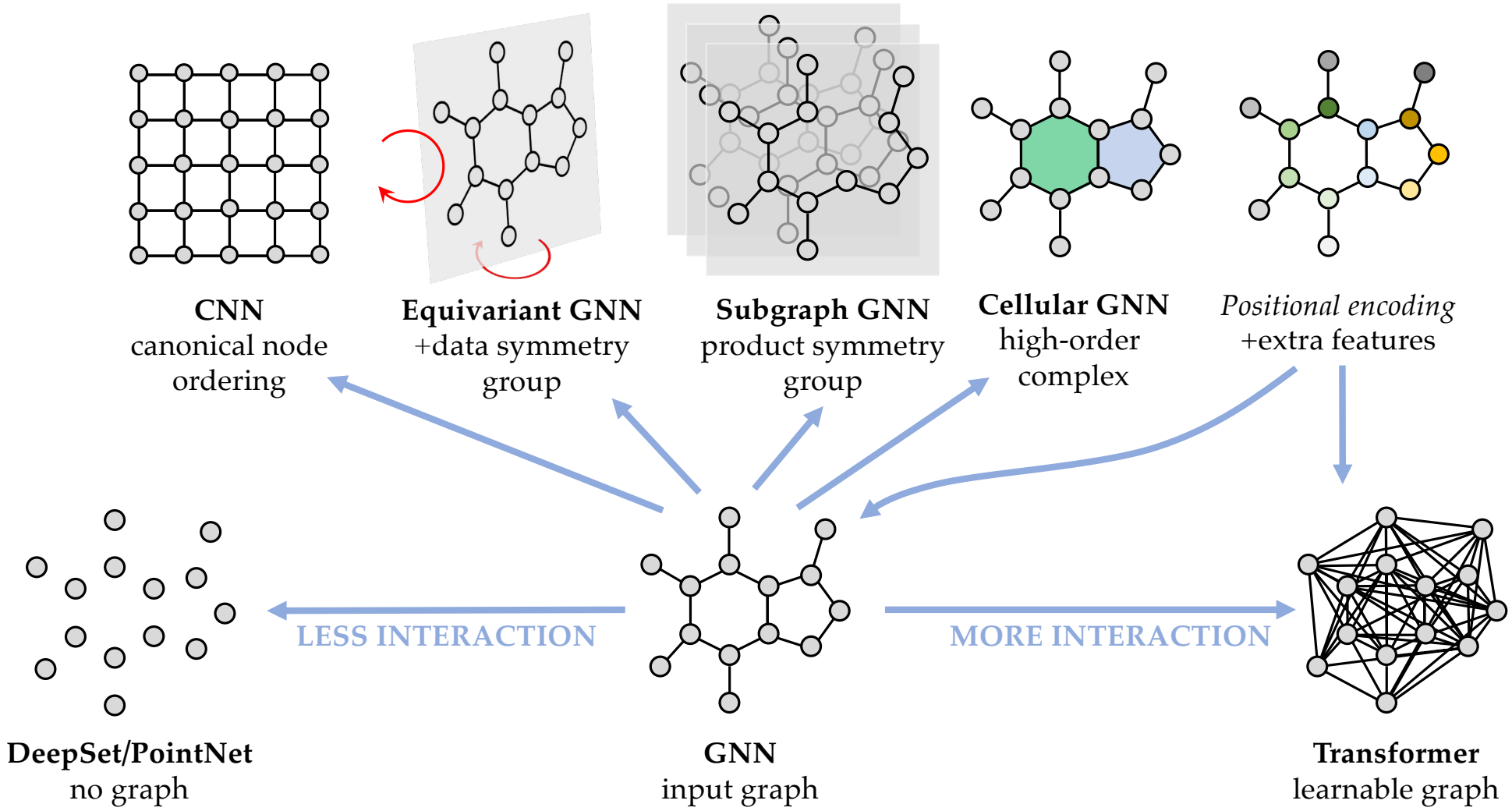
Gap between Theory & Practice

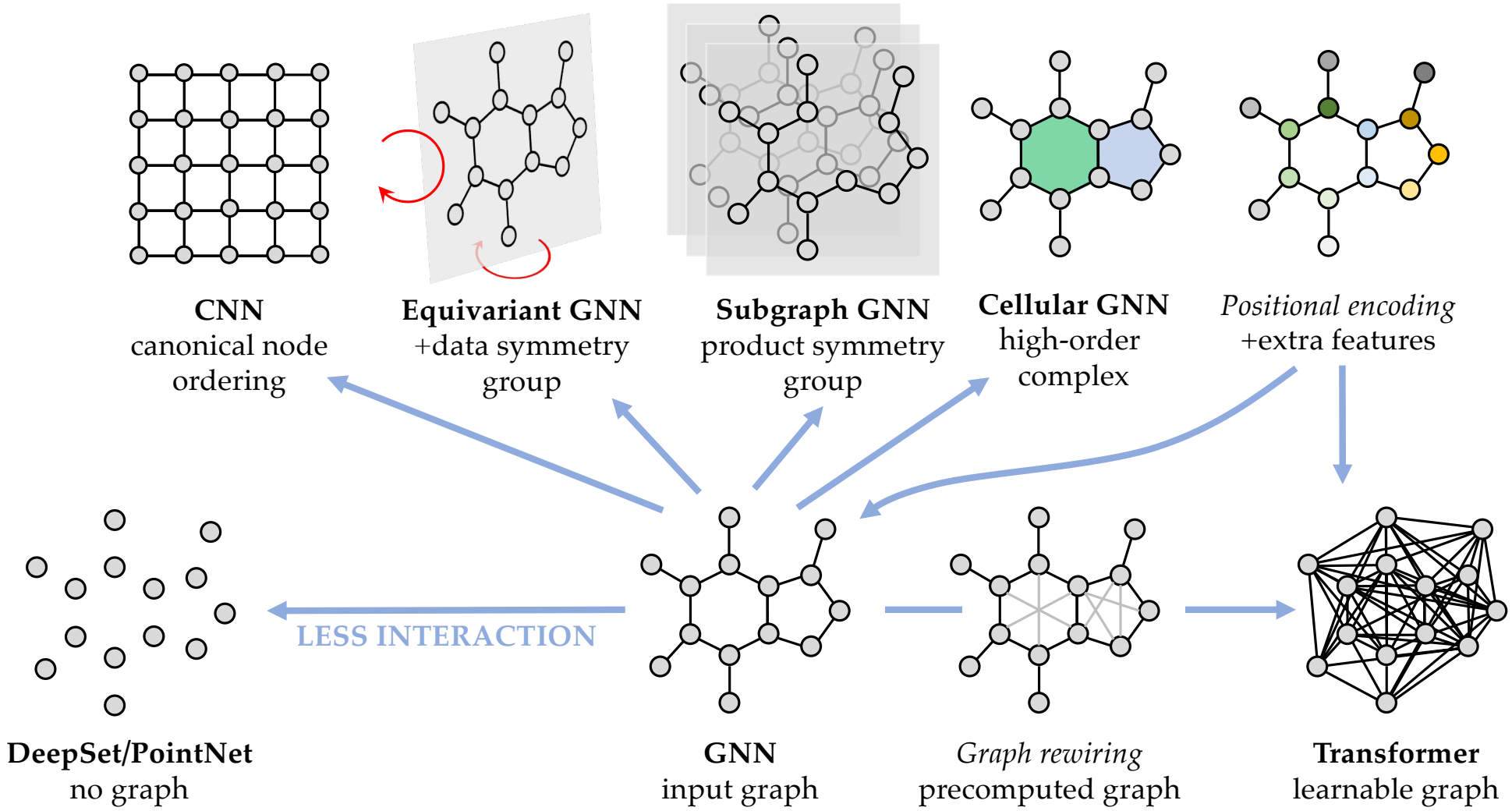
Graphs may be unfriendly for message passing resulting in "bottlenecks"



"Graph rewiring"

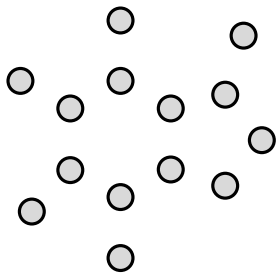
Alon, Yahav 2020 (bottlenecks); Hamilton et al. 2017 (neighbour sampling); Klicpera et al. 2019 (diffusion); Topping, Di Giovanni et B 2022 (Ricci flow); Deac et al. 2022 (expanders)


MORE STRUCTURE
CNN
canonical node ordering
Equivariant GNN
+data symmetry group
Subgraph GNN
product symmetry group
Cellular GNN
high-order complex
Positional encoding
+extra features
LESS INTERACTION
MORE INTERACTION
DeepSet/PointNet
no graph
GNN
input graph
Transformer
learnable graph


MORE STRUCTURE
CNN
canonical node ordering
Equivariant GNN
+data symmetry group
Subgraph GNN
product symmetry group
Cellular GNN
high-order complex
Positional encoding
+extra features
LESS INTERACTION
DeepSet/PointNet
no graph
GNN
input graph
Graph rewiring
precomputed graph
Transformer
learnable graph

TRANSFORMERS

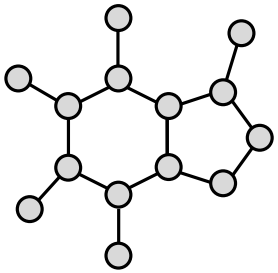
What graph should we use in GNNs?



DeepSet
no graph at all

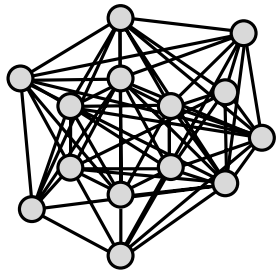
– Throws away important data

LESS INTERACTION



GNN
input graph

MORE INTERACTION



?
Complete graph

+ Learn the "right" graph for the task

– High complexity

– "Too much flexibility"

Transformers

- Assume the graph is *complete* (every node is connected to every node)
- Apply a Convolutional GNN:

$$\phi\left(\mathbf{x}_i, \sum_{j \in \mathcal{N}_i} a_{ij} \psi\left(\mathbf{x}_j\right)\right)$$

Transformers

- Assume the graph is *complete* (every node is connected to every node)
- Apply a Convolutional GNN:

$$\phi\left(\mathbf{x}_i, \sum_{j=1}^{n} a_{ij}\psi(\mathbf{x}_j)\right)$$

equal for every node!

Transformers

- Assume the graph is *complete* (every node is connected to every node)
- Apply an Attentional GNN:

$$\phi\left(\mathbf{x}_i, \sum_{j=1}^{n} \alpha(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$

- Attention weights = *learned graph adjacency* on the downstream task
- **Transformer** architecture, which has recently become predominant in NLP task, is a special type of GNN!

Vaswani et al. 2017; Joshi 2020

Transformers

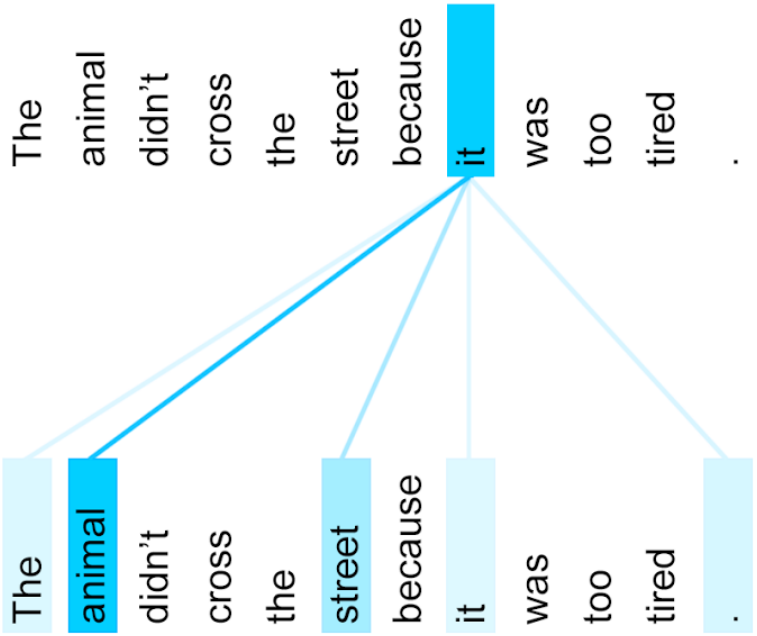
- Assume the graph is *complete* (every node is connected to every node)
- Apply an Attentional GNN:

$$\phi\left(\mathbf{x}_i, \sum_{j=1}^{n} \alpha(\mathbf{x}_i, \mathbf{x}_j, \mathbf{p}_i, \mathbf{p}_j)\psi(\mathbf{x}_j)\right)$$

- Attention weights = *learned graph adjacency* on the downstream task
- **Transformer** architecture, which has recently become predominant in NLP task, is a special type of GNN!
- Tasks in NLP often require *position-dependent* functions (not permutation-invariant), which is achieved through **positional encoding**

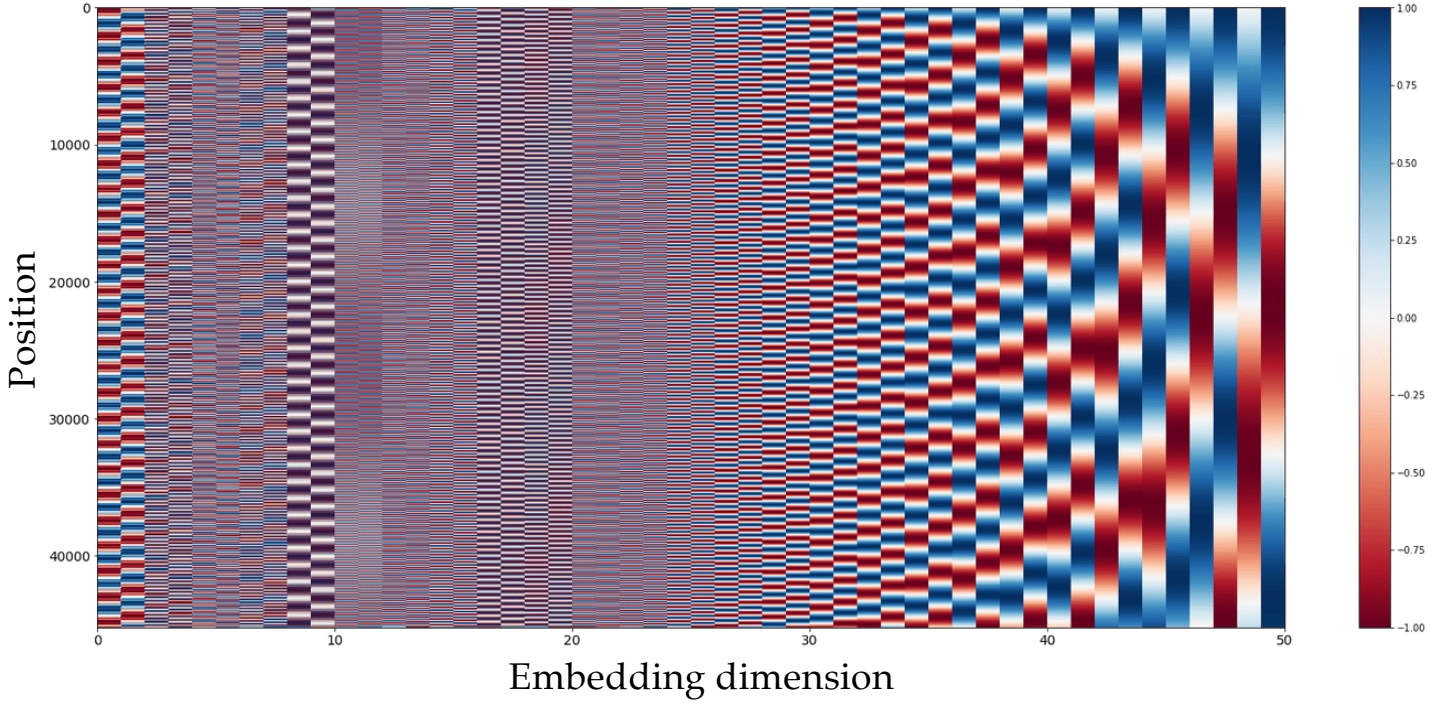
Vaswani et al. 2017; Joshi 2020

Transformers



Vaswani et al. 2017; Joshi 2020

Positional encoding

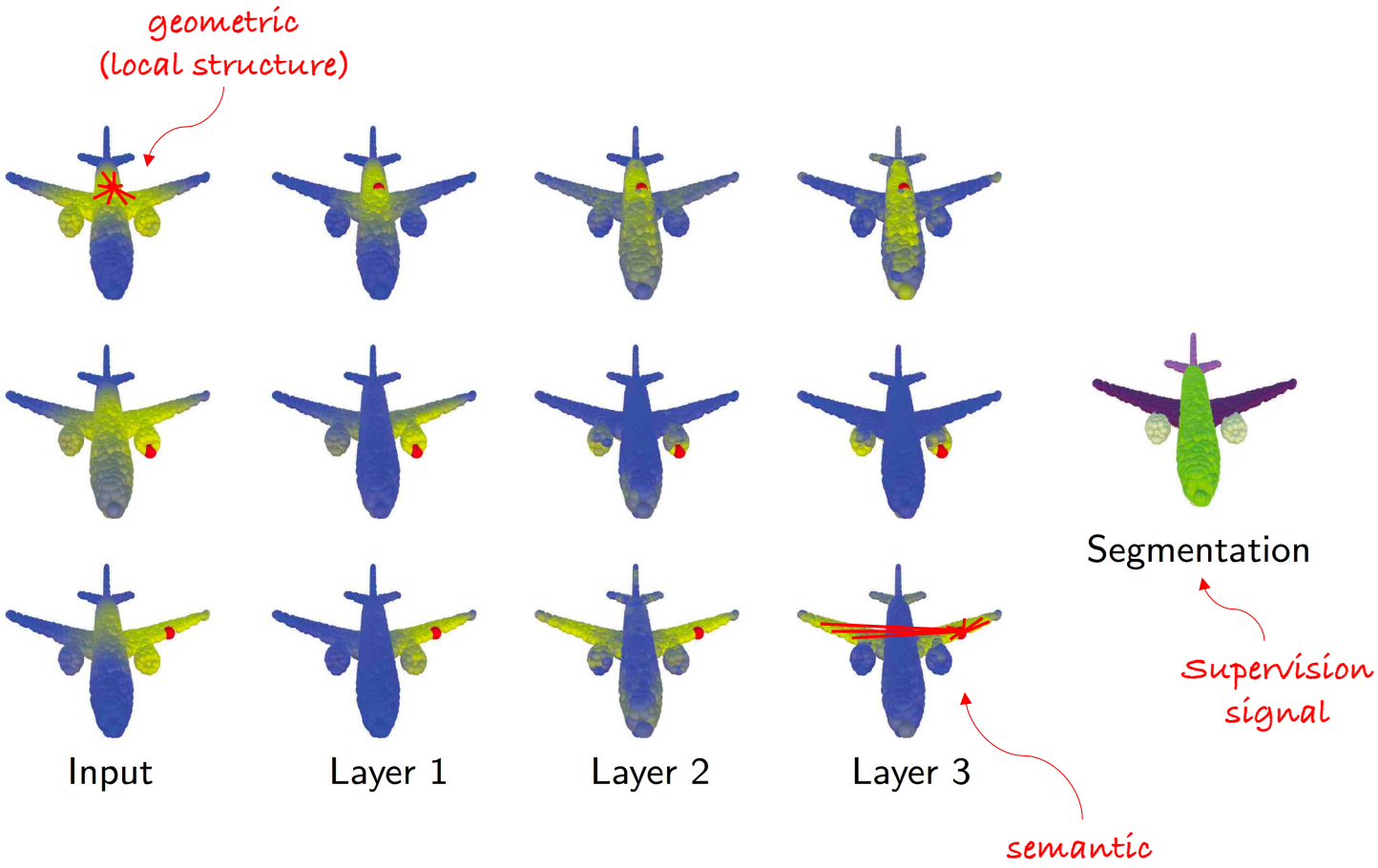


Typical positional encoding used in Transformers on a 1D grid

Positional encoding

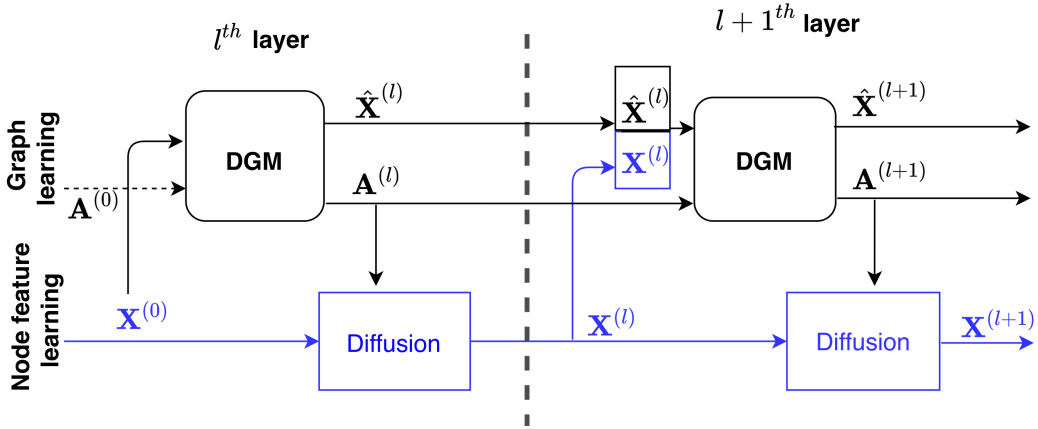
	Unique?	Complexity
Substructures of size k	Yes	$\mathcal{O}(n^k)^*$
k Laplacian eigenvectors	No 2^k sign flips	$\mathcal{O}(n^2) \div \mathcal{O}(n^3)^{**}$
Random Walk kernels	Yes	$\mathcal{O}(n^2) \div \mathcal{O}(n^3)^{**}$

*In practice better for many substructures such as triangles
**Depending on the sparsity of the Laplacian



Wang et B 2018

Differentiable Graph Module



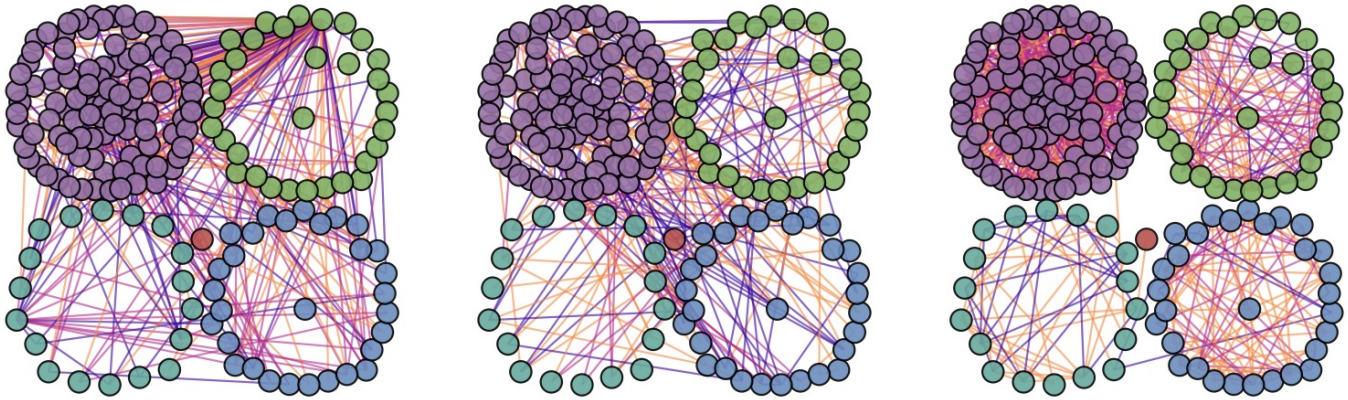
Differentiable Graph Module (DGM) allowing to construct the graph from the data and use it for feature learning

Method	TADPOLE		UK Biobank	
	Transductive	Inductive	Transductive	Inductive
DGCNN	84.59±4.33	**82.99±4.91**	58.35±0.91	**51.84±8.16**
LDS	**87.06±3.67**	†	OOM	†
cDGM	**92.91±2.50**	**91.85±2.62**	**61.32±1.51**	**55.91±3.49**
dDGM	**94.10±2.12**	**92.17±3.65**	**63.22±1.12**	**57.34±5.32**

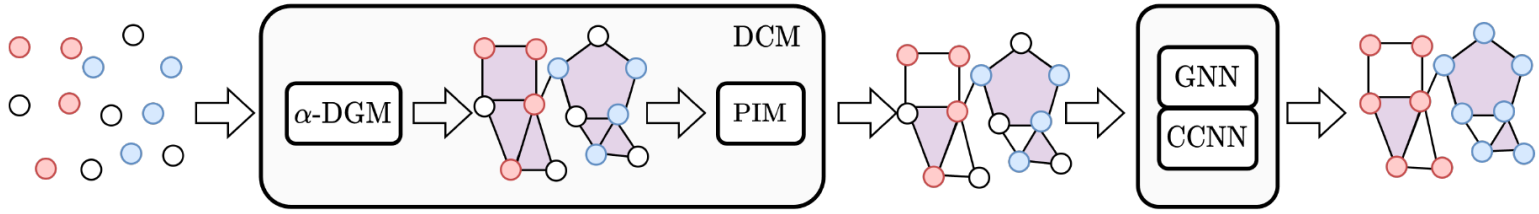
Disease classification accuracy

Kazi, Cosmo et B 2020

Latent Simplicial Complex Learning



(a) Input, $\%_p = 100$, $h = 0.11$ (b) Epoch 40, $\%_p = 90$, $h = 0.4$ (c) Epoch 180, $\%_p = 20$, $h = 0.99$

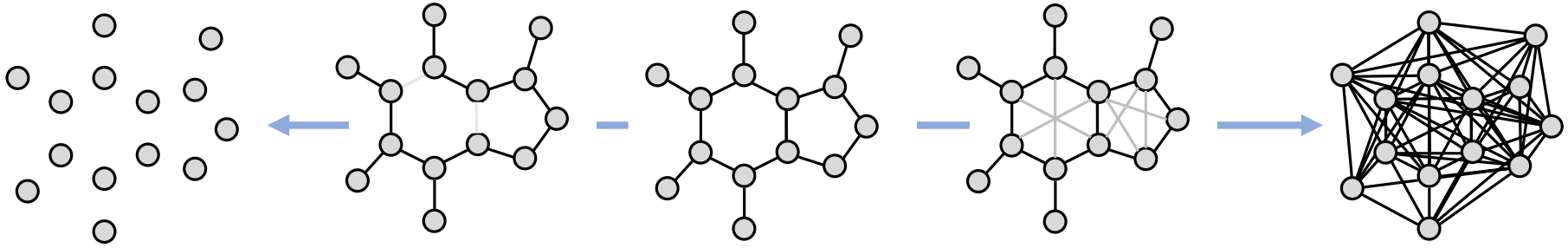


Differentiable Cell Complex Module

Battiloro et B, Scardapane, Di Lorenzo 2024

OVERSQUASHING & GRAPH REWIRING

What graph should we use in GNNs?



DeepSet
no graph at all

– Throws away important data

Classical MPNN
input graph

Graph rewiring
add / remove edges

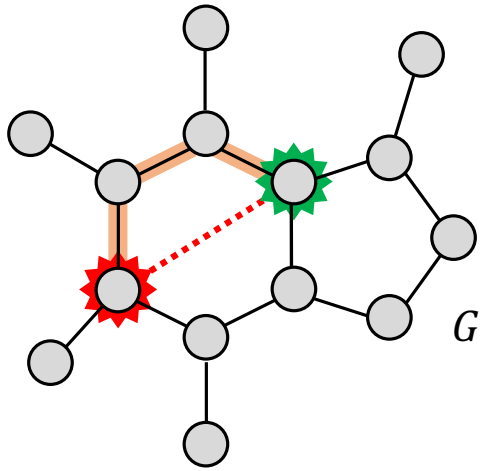
Transformers
complete graph

+ Learn the "right" graph for the task

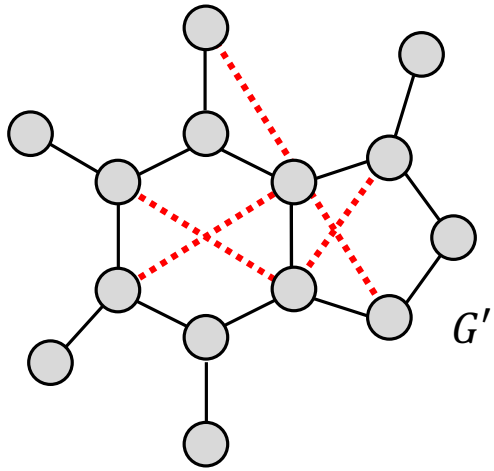
– High complexity

– Hard to generalise

What graph should we use in GNNs?



What graph should we use in GNNs?

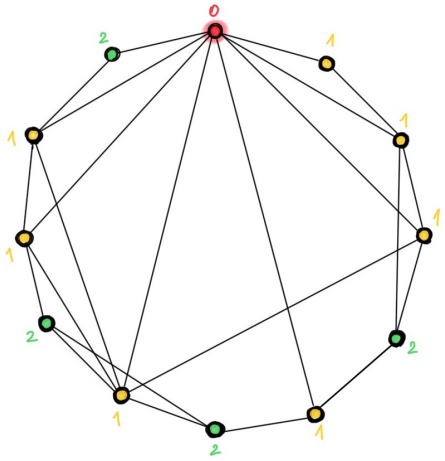


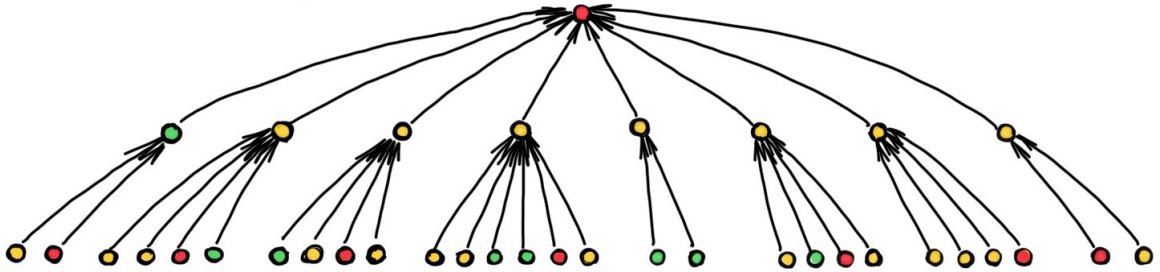
- Preserve locality ("input graph inductive bias")
- Preserve sparsity (computational efficiency)
- Improve connectivity (to reduce over-squashing)

Barbero, Velingker, Saberi, B, Di Giovanni 2023

“Failure of Message Passing to propagate information on the graph”

Over-squashing





In small-world graphs metric ball volume grows exponentially with ball radius

graph topology

Over-squashing = Fast volume growth

\+ Long-range interactions

task

Alon, Yahav 2020

Over-squashing

- Consider an MPNN of the form

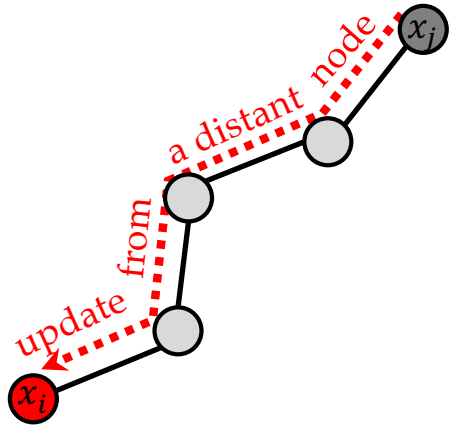
$$\mathbf{x}_i^{(k+1)} = \sigma\left(\mathbf{W}_1\mathbf{x}_i^{(k)} + \sum_j a_{ij}\,\mathbf{W}_2\mathbf{x}_j^{(k)}\right)$$

  - L = *depth* (number of layers)
  - p =*width* (hidden dimension)
  - Nonlinearity σ is c_σ-*Lipschitz-continuous*
  - w = maximum element of weight matrices $\mathbf{W}_1$, $\mathbf{W}_2$

Theorem (Sensitivity bound): For any $i, j \in V$

$$\left\|\frac{\partial \mathbf{x}_i^{(L)}}{\partial \mathbf{x}_j^{(0)}}\right\|_1 \leq (c_\sigma w p)^L (\mathbf{I} + \mathbf{A})^L_{ij}$$

model topology



Over-squashing: small Jacobian $\left\|\partial \mathbf{x}_i^{(L)} / \partial \mathbf{x}_j^{(0)}\right\|$ indicates poor information propagation from input node

Topping, Di Giovanni et B 2021; Di Giovanni et B 2023

Preventing over-squashing

$$\left\| \frac{\partial \mathbf{x}_i^{(L)}}{\partial \mathbf{x}_j^{(0)}} \right\|_1 \leq (c_\sigma w p)^L (\mathbf{I} + \mathbf{A})^L_{ij}$$

model topology

- **Width** p helps mitigate over-squashing (potentially at the risk of worse generalization)
- **Depth** L does not help
  - If $L \sim \text{diam}(G)$, over-squashing occurs between distant nodes
  - If $L \gg 1$, we transition from over-squashing to vanishing gradients
- **Topology** of G has the largest effect on over-squashing, which occurs
  - Between nodes with high *effective resistance* $\text{Res}(i,j) \propto \tau(i,j) = \textit{commute time}$
  - On graphs with small *Cheeger constant* $h(G)$ = energy required to disconnect the graph into two communities)
  - Edges with strongly negative discrete *Ricci curvature* $\kappa(i,j)$

Di Giovanni et B 2023

Effective Resistance & Commute Time

- **Commute time** $\tau(i, j)$ = expected number of steps a random walk on a graph starting from node i will take to reach node j and come back
- **Effective resistance** $\text{Res}(i, j)$ = voltage difference between nodes i and j if a unit current flows through the graph where every edge has unit resistance

$$\tau(i, j) = 2|E|\text{Res}(i, j)$$

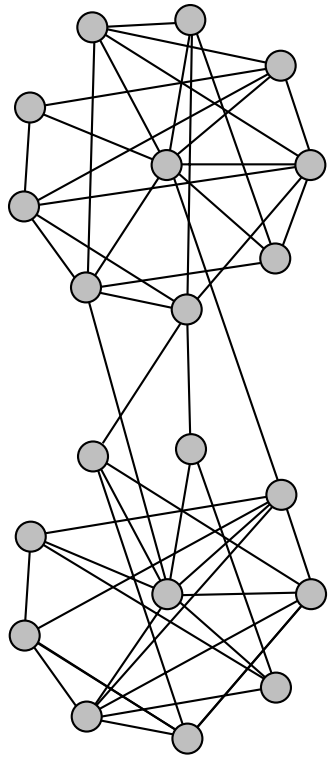
Chandra et al. 1996

Cheeger constant

- The **Cheeger constant** of a graph $G = (V, E)$ is defined as

$$h(G) = \min_{U \subset V} \frac{|\{(i,j) \in E \text{ s.t. } i \in U \text{ and } j \in V \setminus U\}|}{\min\{\text{vol}(U), \text{vol}(V)\}}$$

  where $\text{vol}(U) = \sum_{j \in U} d_i$.



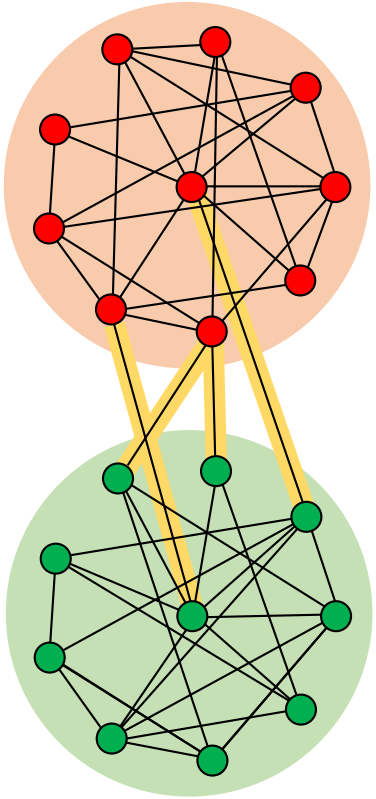
(Cheeger 1970)

Cheeger constant

- The **Cheeger constant** of a graph $G = (V, E)$ is defined as

$$h(G) = \min_{U \subset V} \frac{|\{(i,j) \in E \text{ s.t. } i \in U \text{ and } j \in V \setminus U\}|}{\min\{\text{vol}(U), \text{vol}(V)\}}$$

where $\text{vol}(U) = \sum_{j \in U} d_i$.



(Cheeger 1970)

Cheeger constant

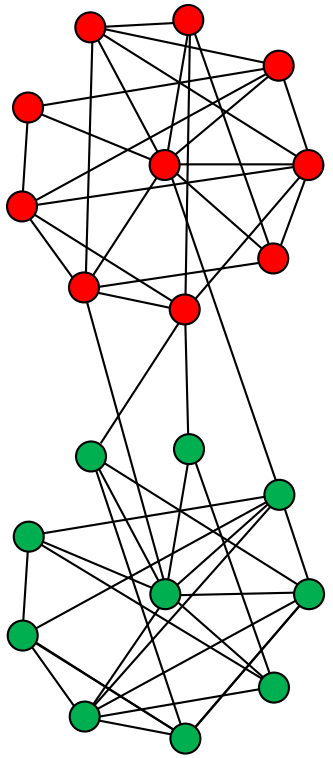
- The **Cheeger constant** of a graph $G = (V, E)$ is defined as

$$h(G) = \min_{U \subset V} \frac{|\{(i,j) \in E \text{ s.t. } i \in U \text{ and } j \in V \setminus U\}|}{\min\{\text{vol}(U), \text{vol}(V)\}}$$

  where $\text{vol}(U) = \sum_{j \in U} d_i$.

- Energy required to disconnect G into separate communities ("bottleneck" if h is small)

- **Cheeger inequality:** $2h(G) \geq \lambda_1 > \frac{h^2(G)}{2}$, where λ_1 is the first non-zero eigenvalue ("spectral gap") of the normalised graph Laplacian

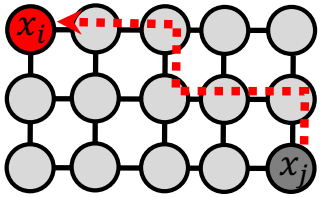


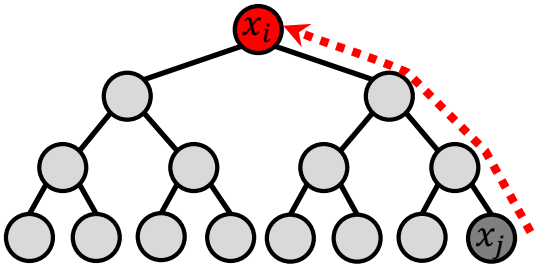
(Cheeger 1970)

Preventing over-squashing

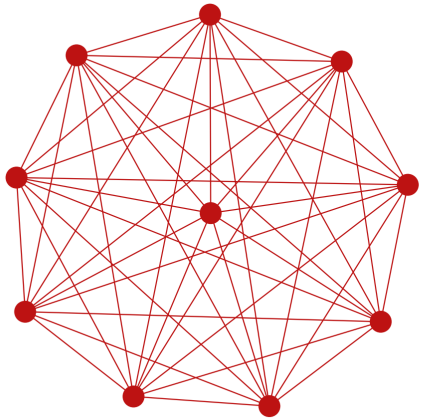
$$\left\| \frac{\partial \mathbf{x}_i^{(L)}}{\partial \mathbf{x}_j^{(0)}} \right\|_1 \leq (c_\sigma w p)^L (\mathbf{I} + \mathbf{A})^L_{ij}$$

model topology

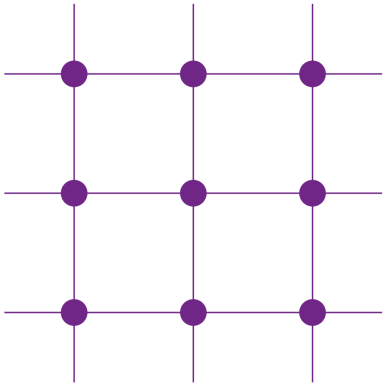




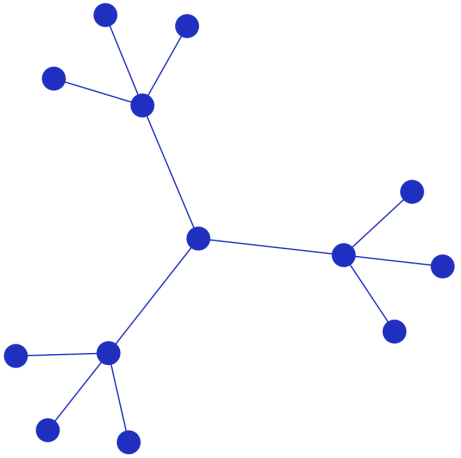
When oversquashing might occur?



Clique

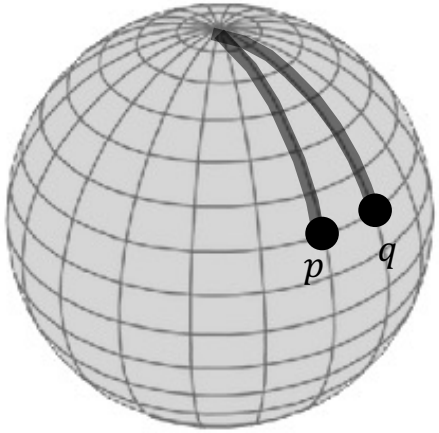


Grid

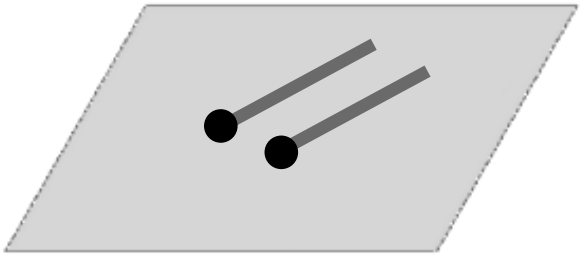


Tree

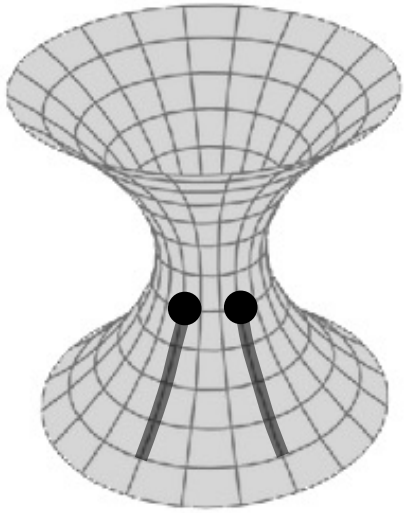
Ricci Curvature on Manifolds



Spherical (>0)



Euclidean (=0)

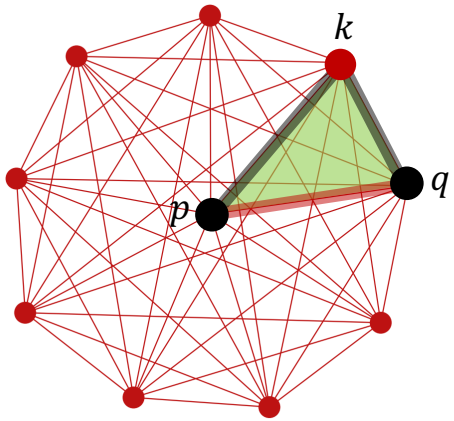


Hyperbolic (<0)

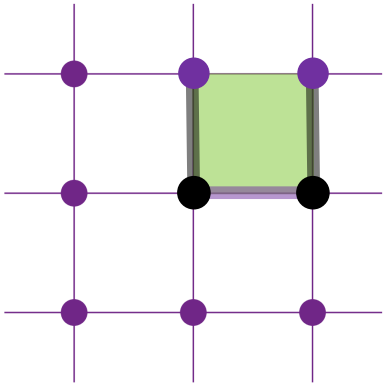
"geodesic dispersion"

Ricci 1903

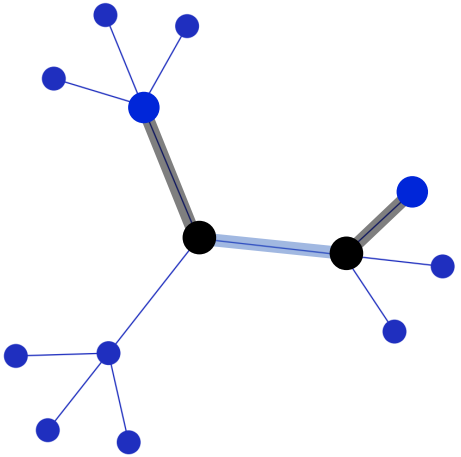
Graph analogues of Ricci curvature



Clique (>0)



Grid (=0)



Tree (<0)

Graph Ricci curvature

- Multiple definitions of curvature on graphs, two main
  - Forman (combinatorial)
  - Ollivier (optimal transport)
- **Balanced Forman Curvature** of edge $i \sim j$ in simple unweighted graph is
  - $\kappa(i,j) = 0$ if $\min\{d_i, d_j\} = 1$, otherwise
  - $$\kappa(i,j) = \frac{2}{d_i} + \frac{2}{d_j} + 2\frac{|\sharp_\Delta(i,j)|}{\max\{d_i,d_j\}} + \frac{|\sharp_\Delta(i,j)|}{\min\{d_i,d_j\}} + \frac{\gamma_{\max}^{-1}}{\max\{d_i,d_j\}}\left(\left|\sharp_\square^i(i,j)\right| + \left|\sharp_\square^j(i,j)\right|\right) - 2$$

Degree of i

Triangles based at $i \sim j$

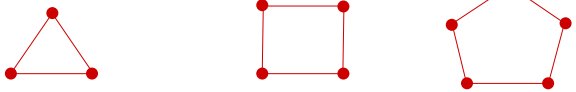
Max number of 4-cycle based at $i \sim j$ traversing the same node

Neighbours of i forming a 4-cycle based at $i \sim j$ (w/o diagonals)

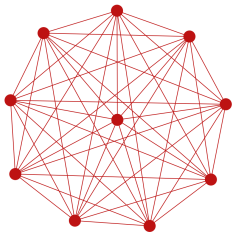
Forman 2003; Ollivier 2007; Topping, di Giovanni, et B. 2021

Graph Ricci curvature

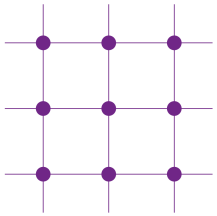
- Multiple definitions of curvature on graphs, two main
  - Forman (combinatorial)
  - Ollivier (optimal transport)
- **Balanced Forman Curvature** of edge $i \sim j$ in simple unweighted graph behaves similarly to the continuous Ricci curvature



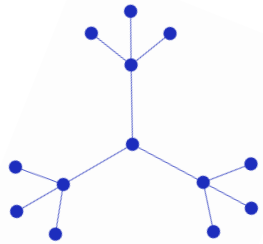
Cycle C_3: $\frac{3}{2}$ C_4: 1 $C_{n\geq 5}$: 0



Clique K_n: $\frac{n}{n-1}$



Grid G_n: 0

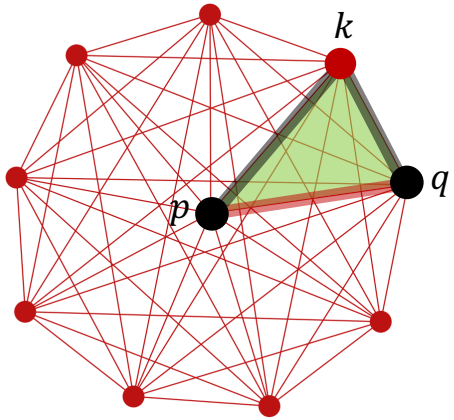


Tree T_r: $\frac{4}{r+1} - 2$

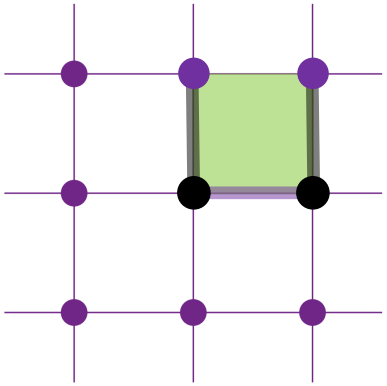
Forman 2003; Ollivier 2007; Topping, di Giovanni, et B. 2021

What contributes to over-squashing?

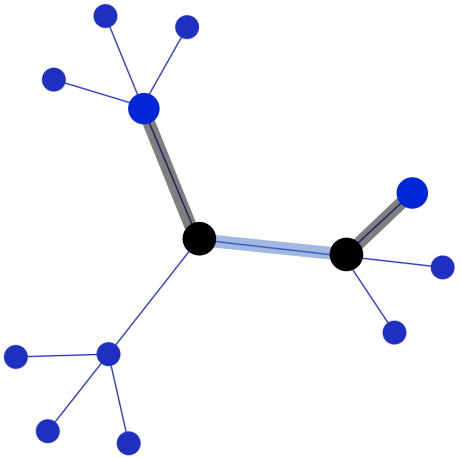
Theorem: (informal) *strong negatively-curved edges* contribute to over-squashing.



Clique (>0)



Grid (=0)



Tree (<0)

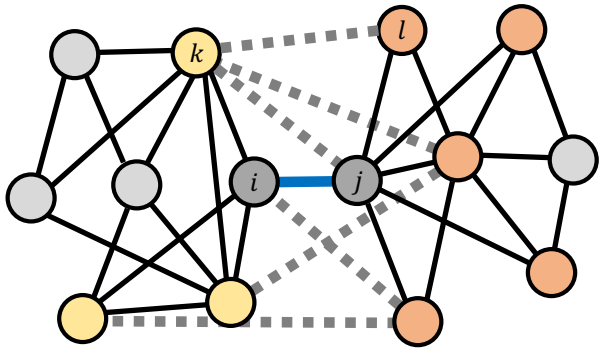
Topping, Di Giovanni et B 2021; *No contradiction to using expander graphs (which have negative curvature)

Curvature-based rewiring

Input: graph $G = (V, E)$, temperature $\tau > 0$, (optional C)

- For edge $i \sim j$ with smallest $\kappa(i, j)$
  - Calculate the improvement $\delta_{kl} = \kappa_{G'}(i, j) - \kappa(i, j)$ from adding edge $k \sim l$ with $k \in B_1(i)$ and $l \in B_1(j)$
  - Sample index k, l with probability $\mathrm{Softmax}(\tau\delta_{kl})$ and add edge $k \sim l$ to E'
- (optional) Remove edge $i \sim j$ with largest $\mathrm{Ric}(i, j) > C$

Output: new graph $G' = (V, E')$

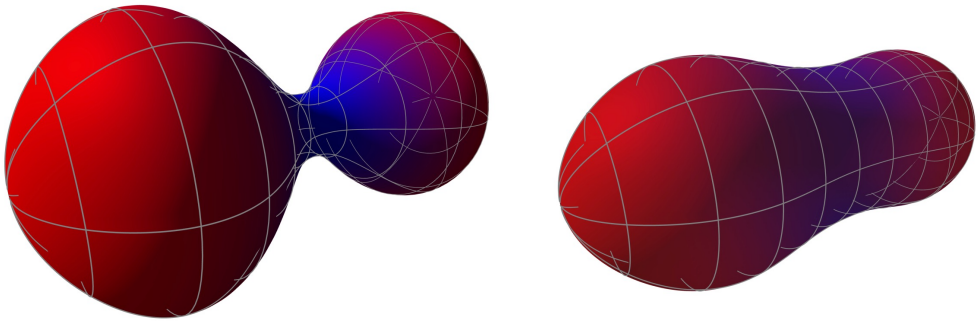


Topping, di Giovanni, et B. 2021

Ricci flow

- **Ricci flow:** "diffusion of the Riemannian metric"

$$\frac{\partial g_{ij}}{\partial t} = R_{ij}$$



Evolution of a manifold under Ricci flow

G. Ricci-Curbastro

R. Hamilton

Ricci 1903; Hamilton 1988;

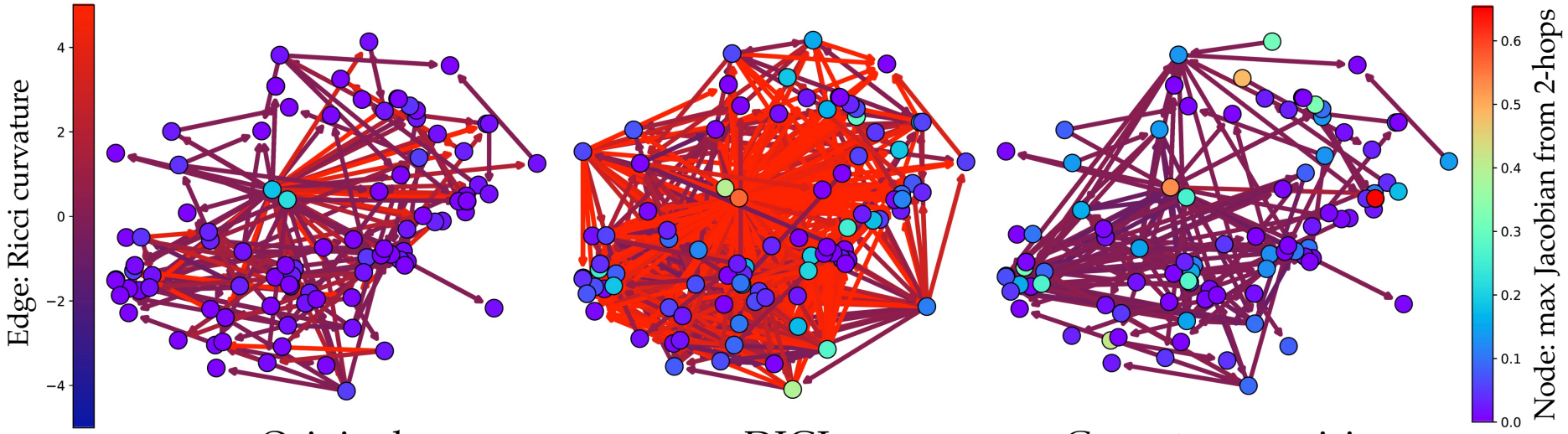
G. Perelman

G. Ricci-Curbastro

R. Hamilton

Ricci 1903; Hamilton 1988; Perelman 2003

Curvature- vs Diffusion-based Rewiring

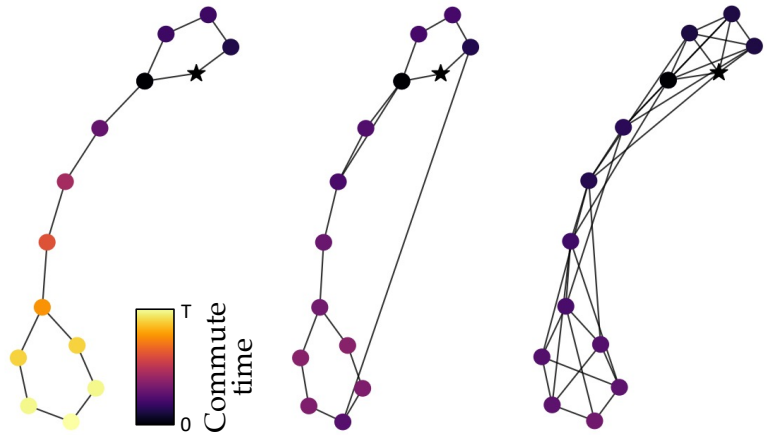


Topping, di Giovanni, et B. 2021; Klicpera et al. 2019 (DIGL)

What contributes to over-squashing?

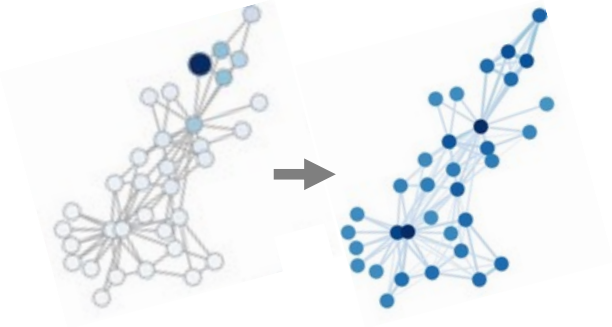
Theorem: (informal) large *commute-time distances* contribute to over-squashing.

- **Spectral rewiring:**[1,2] increase the Cheeger constant of the graph ("clusteredness"), which leads to lower commute time
- **Spatial rewiring:**[3-5] inserting edges reduces the total effective resistance of the graph (=commute-time distance up to scale)
- **Bounded-degree expanders**[6] have commute time $O(|E|)$; over-squashing effect is independent on the size of the graph



Di Giovanni, Giusti, Barbero, Luise, Lio', B 2023
[1]Arnaiz-Rodriguez et al. 2022 (DiffWire); [2]Karhadkar et al. 2022; [3]Abboud et al. 2022; [4]Mialon et al. 2022; [5]Abu-El-Haija et al. 2019; [3]Deac et al. 2022 (EGP) *&* many more

Graph Rewiring Approaches



Connectivity Diffusion
(DIGL)

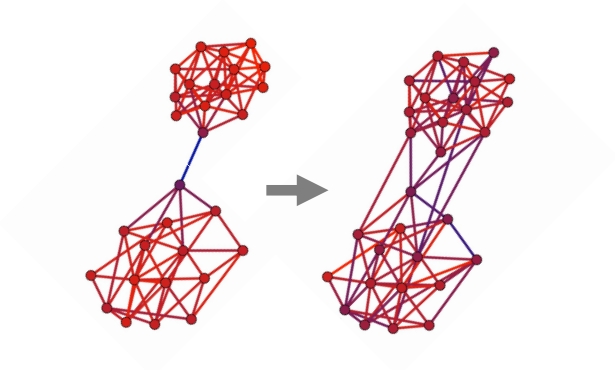
Gasteiger et al. 2019

- KNN graph on PPR embedding

+ Good for homophilic graphs

– Bad for heterophilic graphs

– Drastically different graph



Discrete Ricci flow
(SDRF)

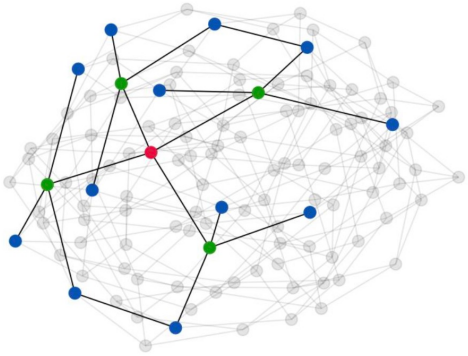
Topping, Di Giovanni et B. 2022

- Remove negatively-curved edges

+ Good for both homophilic and heterophilic graphs

+ "Surgical" rewiring

– Curvature is computationally expensive



Expander Propagation
(EGP)

Deac et al. 2022

- Fixed expander graph
- Alternate MP on original and new graph

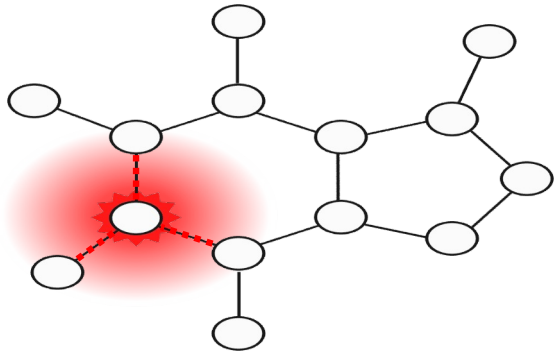
+ Optimal graph for MP

– No relation to input graph

– No permutation equivariance

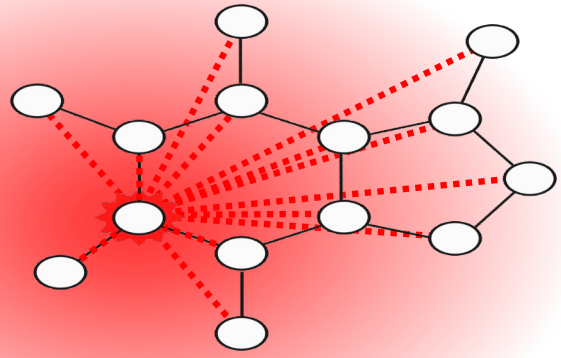
No relation to the task!

Why it is important to consider the task?



Van der Waals interactions

$\propto r^{-12}$



Coulomb interactions

$\propto r^{-1}$

Same graph+features, different task

Whether the graph is good depends on the task!

LONG-RANGE INTERACTIONS & EXPRESSIVE POWER

Long-range interactions in graph tasks

- *Task* = a function $f(\mathbf{X})$ on the node features of a graph G
- The interaction between features in nodes i and j required for the task is given by

$$\textbf{Mixing of } f\text{:} \quad \mathrm{mix}_f(i,j) = \max_{\mathbf{X}} \max_{1\le\alpha,\beta\le d} \left| \frac{\partial^2 f(\mathbf{X})}{\partial x_i^\alpha \partial x_j^\beta} \right|$$

  - $f(\mathbf{X}) = \phi(\mathbf{x}_i) + \phi(\mathbf{x}_j)$ is fully separable, thus $\mathrm{mix}_f(i,j) = 0$
  - $f(\mathbf{X}) = \phi(\langle \mathbf{x}_i, \mathbf{x}_j \rangle)$ mixing depends on how non-linear ϕ is

Di Giovanni, Rusch, B, Deac, Lackenby, Mishra, Veličković 2023

Capacity bounds

$$\underset{\text{task}}{\mathrm{mix}_f}(i,j) \leq \sum_{k=1}^{\boldsymbol{L}-1} \underbrace{(\boldsymbol{c}_{\sigma}\boldsymbol{W})^{2^{\boldsymbol{L}-k-1}}}_{\text{model}} \underbrace{\left(\boldsymbol{W}\left(\mathbf{S}^{\boldsymbol{L}-k}\right)^{\mathrm{T}}\mathrm{diag}\left(\mathbf{1}^{\mathrm{T}}\mathbf{S}^{k}\right)\mathbf{S}^{\boldsymbol{L}-k} + \boldsymbol{C}\mathbf{Q}_k\right)_{ij}}_{\text{topology}}$$

What is the capacity of MPNN required for a given task?

model + topology mixing

Di Giovanni, Rusch, B, Deac, Lackenby, Mishra, Veličković 2023

Capacity bounds

$$\mathrm{mix}_f(i,j) \leq \sum_{k=1}^{L-1} (c_\sigma w)^{2L-k-1} \left(w \left(\mathbf{S}^{L-k} \right)^{\mathrm{T}} \mathrm{diag}\left(\mathbf{1}^{\mathrm{T}} \mathbf{S}^k \right) \mathbf{S}^{L-k} + C \mathbf{Q}_k \right)_{ij}$$

Bound on weights w

$$w \geq \frac{d_{\min}}{c_2} \left(\frac{\mathrm{mix}_f(i,j)}{q} \right)^{1/d(i,j)}$$

- $d_{\min}$ =min node degree
- Fixed depth $L = \lceil d(i,j)/2 \rceil$
- q =number of paths of length $d(i,j)$ between i and j

Bound on depth L

$$L \geq \frac{\tau(i,j)}{4c_2} + \frac{|E|}{\sqrt{d_i d_j}} \left(\alpha \mathrm{mix}_f(i,j) - \beta \right)$$

- d_i =degree of node i
- α, β =model-related constants
- $|E|$ =number of edges
- Bounded weights

Di Giovanni, Rusch, B, Deac, Lackenby, Mishra, Veličković 2023

Capacity bounds

$$\mathrm{mix}_f(i,j) \leq \sum_{k=1}^{L-1} (c_\sigma w)^{2L-k-1} \left(w \left(\mathbf{S}^{L-k}\right)^{\mathrm{T}} \mathrm{diag}\left(\mathbf{1}^{\mathrm{T}} \mathbf{S}^k\right) \mathbf{S}^{L-k} + C\mathbf{Q}_k \right)_{ij}$$

Bound on weights w

$$w \geq \frac{d_{\min}}{c_2} \left(\frac{\mathrm{mix}_f(i,j)}{q} \right)^{1/d(i,j)}$$

"weights need to be large enough to allow mixing"

Bound on depth L

$$L \geq \frac{\tau(i,j)}{4c_2} + \frac{|E|}{\sqrt{d_i d_j}} \left(\alpha \mathrm{mix}_f(i,j) - \beta \right)$$

- *Depth must be ~commute time $\tau(i,j)$*
- *Rewiring tries to improve τ*
- *τ can be as large as $O(n^3)$, which implies impossibility statements*

Di Giovanni, Rusch, B, Deac, Lackenby, Mishra, Veličković 2023

New way of characterising expressive power

Expressive power (informal): MPNN with $L \leq n$ layers cannot learn tasks that require high mixing among features at nodes with large commute time.

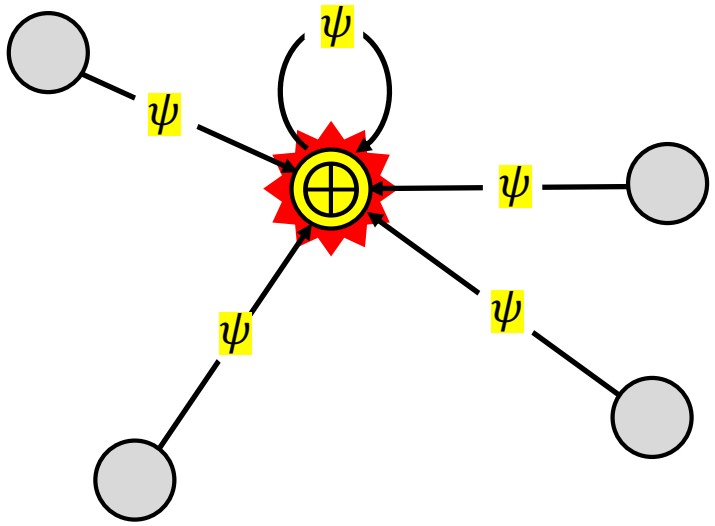
A. Lehman

B. Weisfeiler

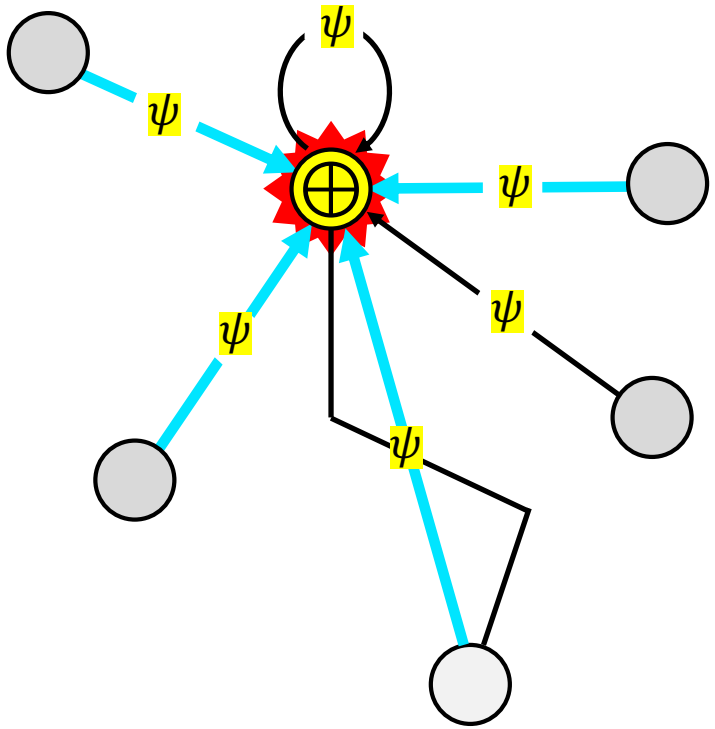
Di Giovanni, Rusch, B, Deac, Lackenby, Mishra, Veličković 2023

EXOTIC MESSAGE PASSING

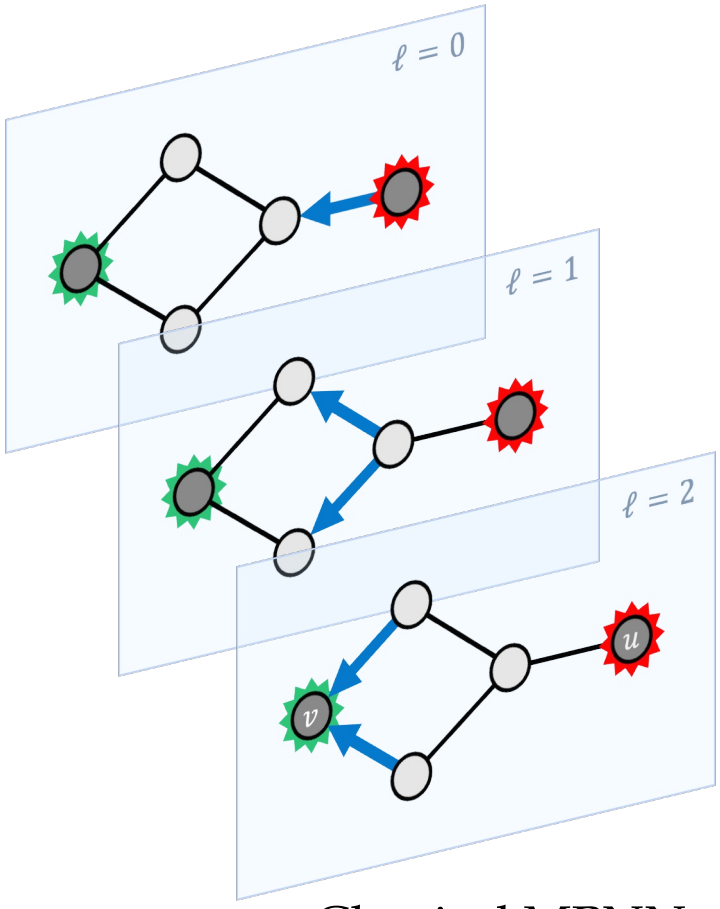
What



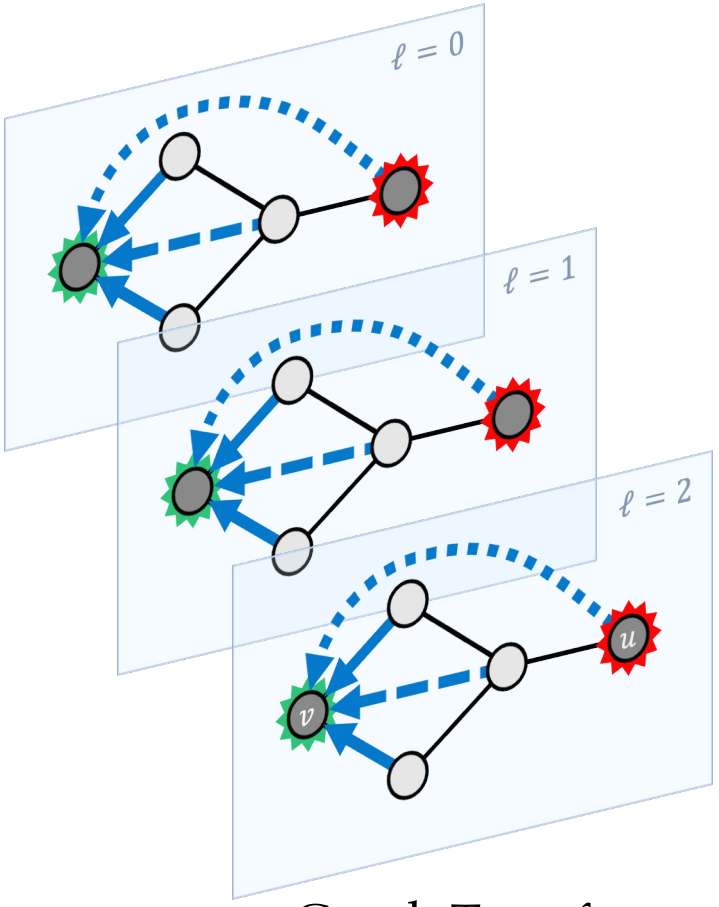
What + Where



What + Where + When

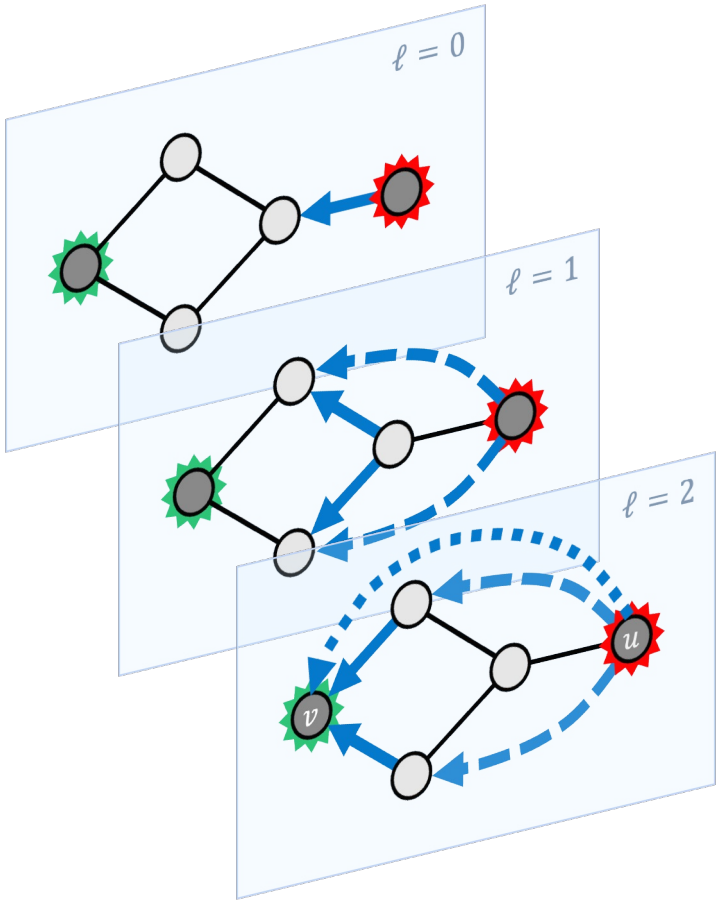


Classical MPNN

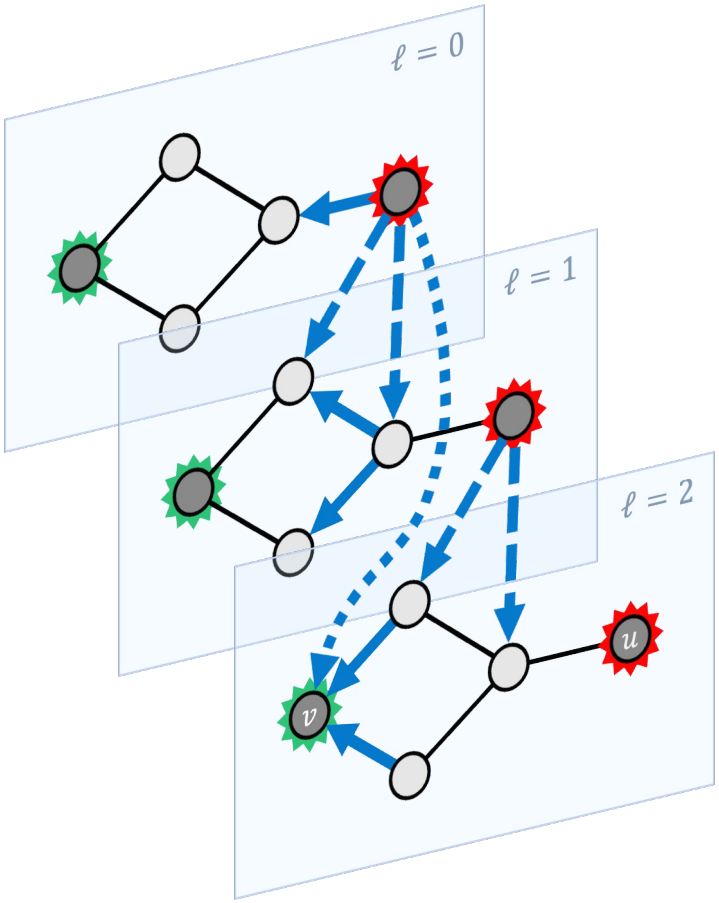


Graph Transformer

Gutteridge, Di Giovanni et B 2023



Dynamic Rewiring (DRew)



Dynamic Rewiring + delay (**v**DRew)

Gutteridge, Di Giovanni et B 2023

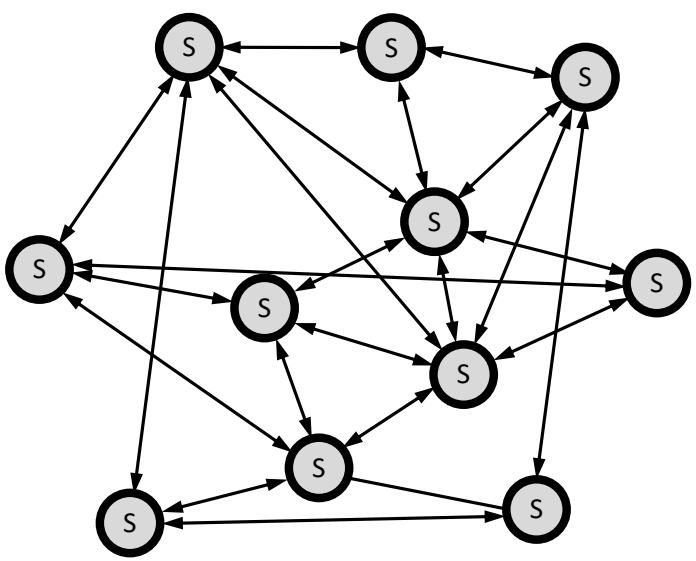
Experimental validation

Model	Peptides-func	Peptides-struct	PCQM-Contact	PascalVOC-SP
	AP ↑	MAE ↓	MRR ↑	F1 ↑
Classical MPNN	0.6069±0.0035	0.3357±0.0006	0.3242±0.0008	0.2860±0.0085
Rewiring+MPNN (DIGL)	0.6830±0.0026	0.2616±0.0018	0.1707±0.0021	0.2921±0.0038
Multi-hop (MixHop-GCN)	**0.6843±0.0049**	0.2614±0.0023	0.3250±0.0010	0.2506±0.0133
Graph Transformer (GT)	0.6326±0.0126	**0.2529±0.0016**	0.3174±0.0020	0.2694±0.0098
GraphGPS (GT+MPNN)	0.6535±0.0041	**0.2500±0.0005**	**0.3337±0.0006**	**0.3748±0.0109**
νDRew-MPNN	**0.7150±0.0044**	**0.2536±0.0015**	**0.3442±0.0006**	**0.3314±0.0024**

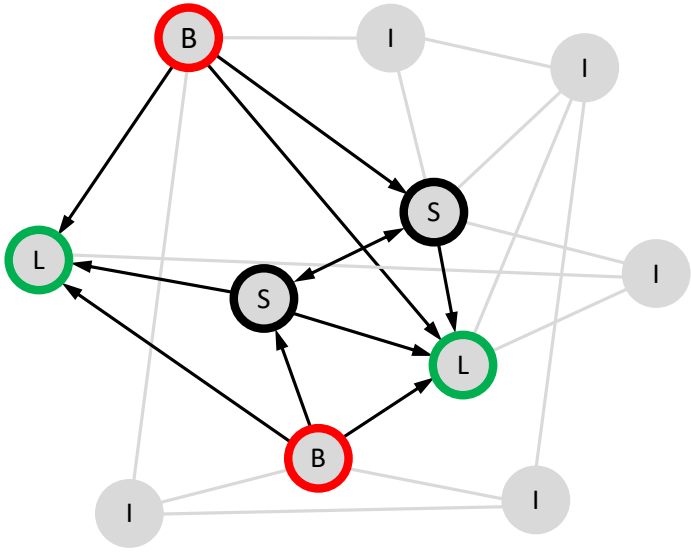
Evaluation on Long-Range Graph Benchmark

Gutteridge et B 2023

Cooperative Message Passing



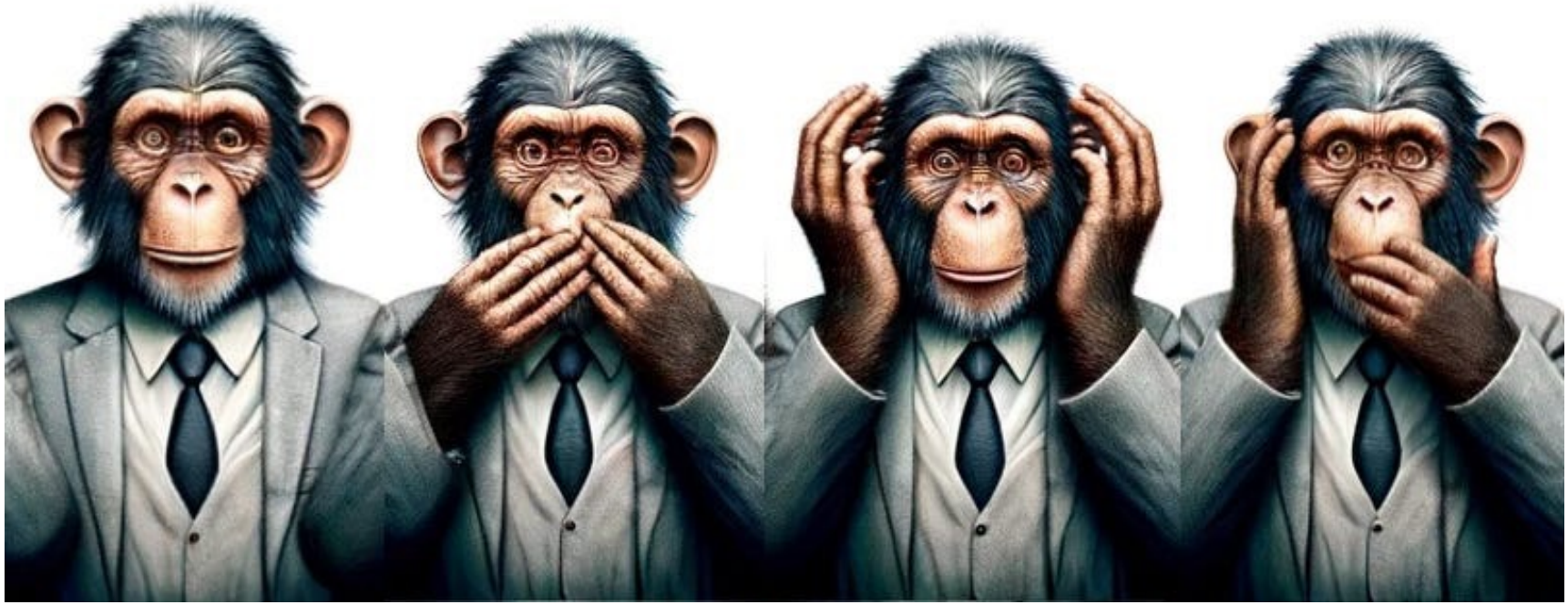
Standard Message Passing
each node Broadcasts & Listens



Cooperative Message Passing
each node individually decides

Finkelshtein, Huang, B, Ceylan 2023

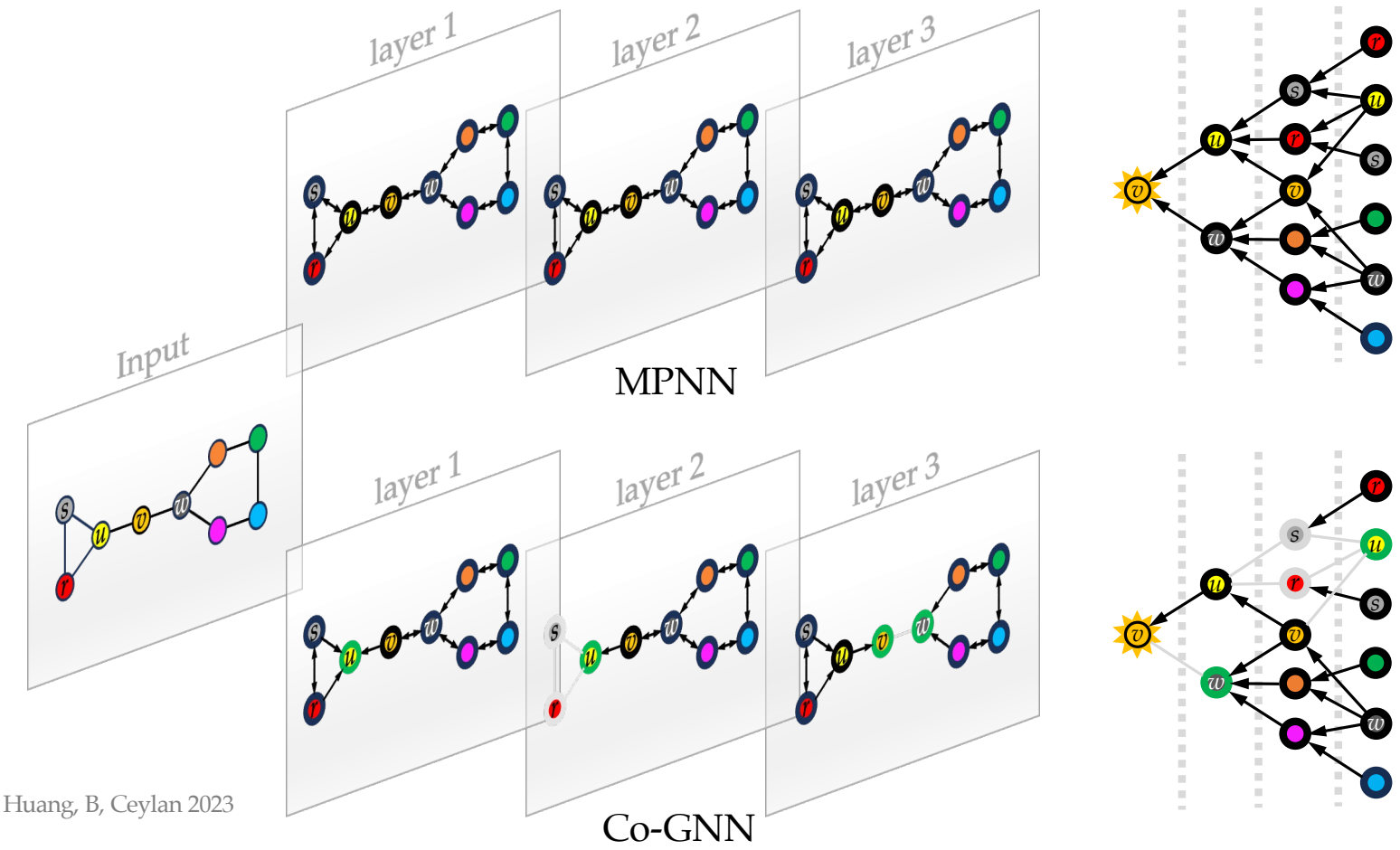
Cooperative Message Passing



Broadcast & Listen | **Listen** | **Broadcast** | **Isolate**

Finkelshtein, Huang, B, Ceylan 2023

Cooperative Message Passing



Finkelshtein, Huang, B, Ceylan 2023

Takeaways

- Input graph is not necessarily the best choice for message passing
- *Graph rewiring* separates input & computational graphs in attempt to avoid *over-squashing*
- Various rewiring techniques based on
  - Discrete Ricci curvature
  - Commute time / resistance distances
  - Spectral properties such as ”connectedness”
- The graph structure impacts the expressive power of MPNN, which can be captured through the *mixing ability* of the GNN
- Advanced rewiring: *dynamic* (DReW) or *implicit* (cooperative GNNs)

Key Concepts

- Transformers and latent graph learning
- Graph rewiring
- Graph Ricci curvature
- Commute time
- Dynamic rewiring
- Co-operative GNNs

Main References

- C. Joshi, Transformers are Graph Neural Networks, 2021
- M. Bronstein, Beyond Message Passing: a Physics-Inspired Paradigm for Graph Neural Networks, *The Gradient* 2022. blog post on various exotic versions of message passing
- M. Bronstein, Graph Neural Networks through the lens of Differential Geometry and Algebraic Topology, *Towards Data Science* 2022.
- M. Bronstein, Over-squashing, Bottlenecks, and Graph Ricci curvature, *Towards Data Science* 2022.
- M. Bronstein, Dynamically Rewired Delayed Message Passing GNNs, *Towards Data Science* 2023.
- M. Bronstein, Co-operative Graph Neural Networks, *Towards Data Science* 2023.