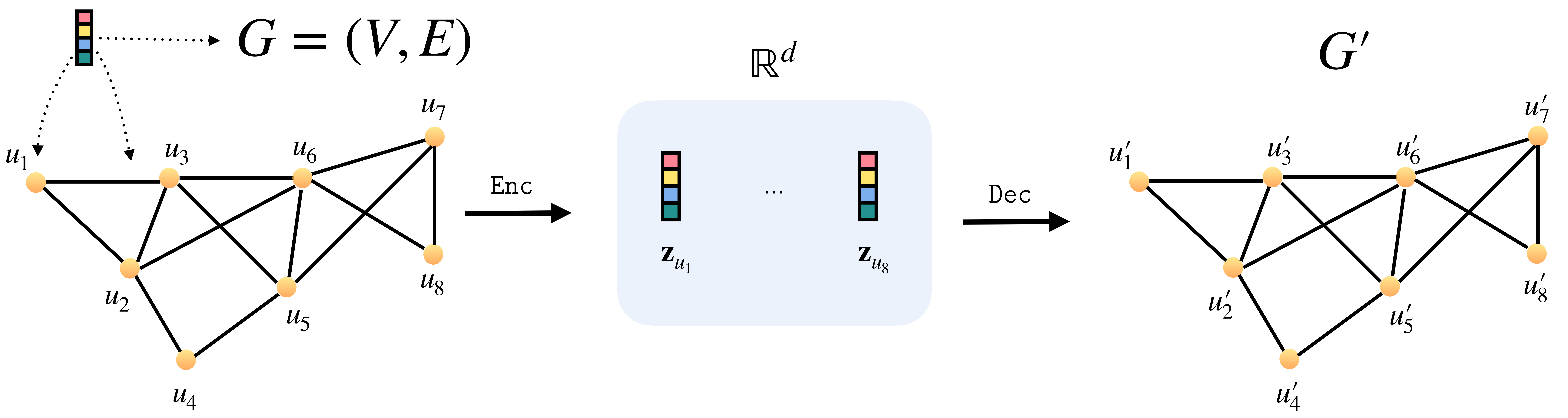
Lecture 6: Higher-Order Graph Neural Networks

Relational Learning

İsmail İlkan Ceylan Advanced Topics in Machine Learning, University of Oxford 26.02.2021

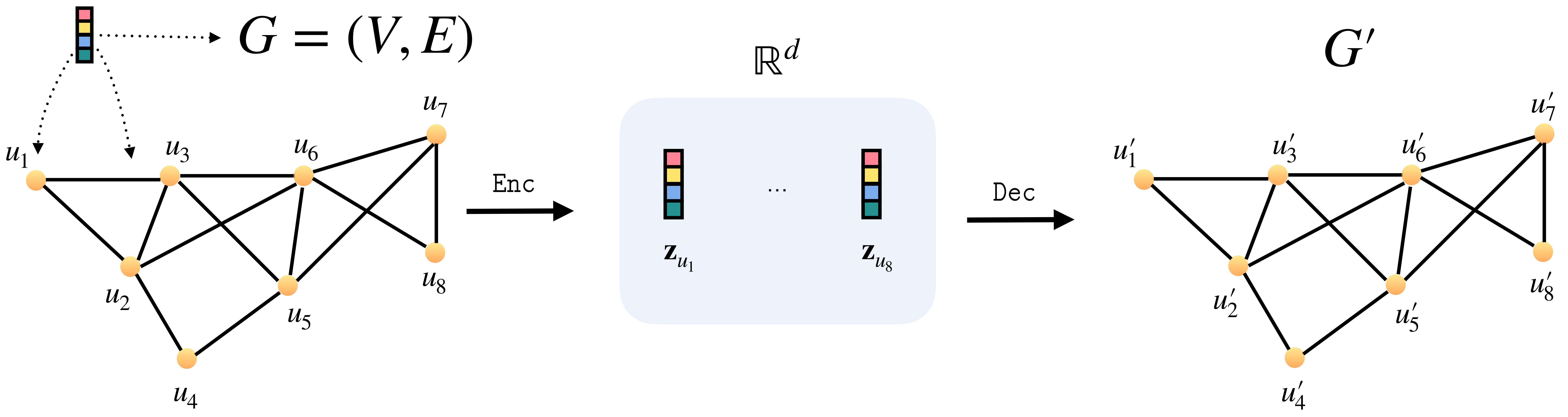
Encoder-Decoder



Our focus so far was on MPNNs

$$\mathbf{h}_u^{(t)} = \mathit{combine}^{(t)}\left(\mathbf{h}_u^{(t-1)}, \mathit{aggregate}^{(t)}\left(\left\{\mathbf{h}_v^{(t-1)} \mid v \in N(u)\right\}\right)\right)$$

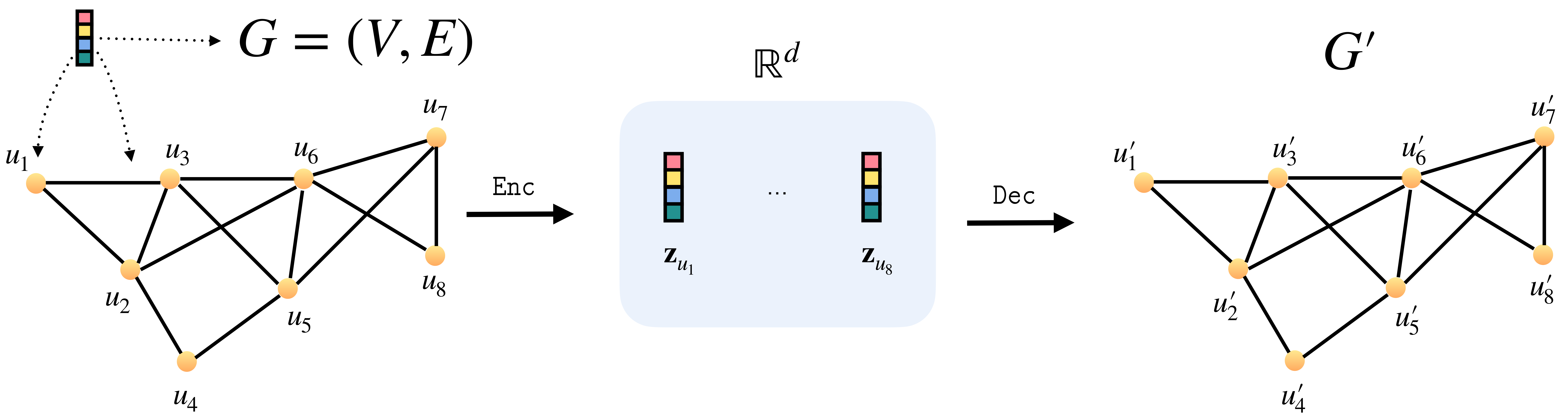
Encoder-Decoder



...and on popular instances of MPNNs

$$\mathbf{h}_u^{(t)} = GRU\left(\mathbf{h}_u^{(t-1)}, \sum_{v \in N(u)} \mathbf{W}^{(t)} \mathbf{h}_v^{(t-1)}\right)$$

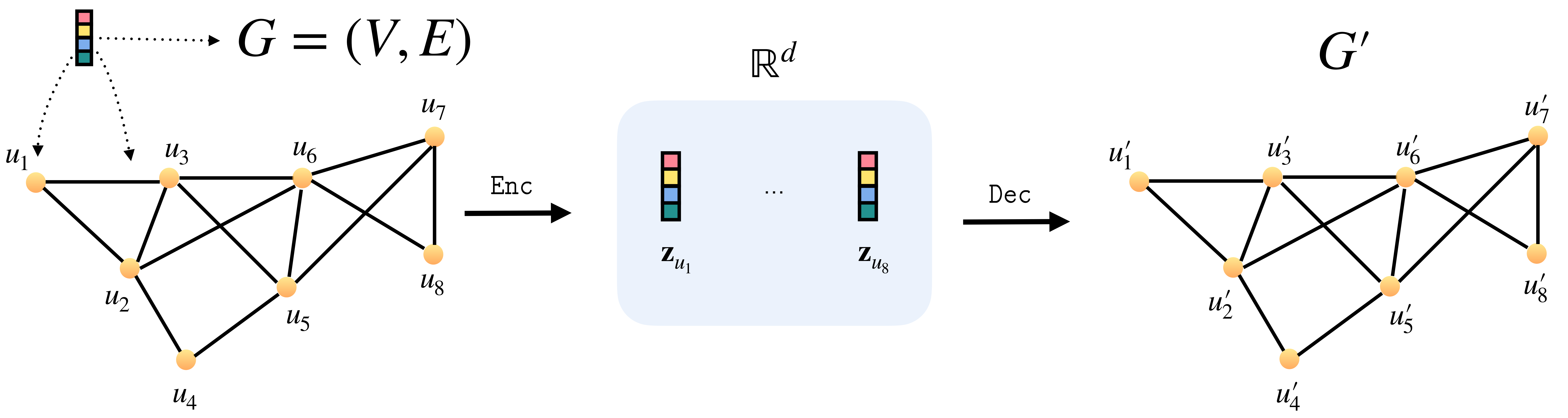
Encoder-Decoder



...and on popular instances of MPNNs

$$\mathbf{h}_u^{(t)} = \sigma\left(\mathbf{W}^{(t)} \sum_{v \in N(u) \cup \{u\}} \frac{\mathbf{h}_v^{(t-1)}}{\sqrt{N(u) + N(v)}}\right)$$

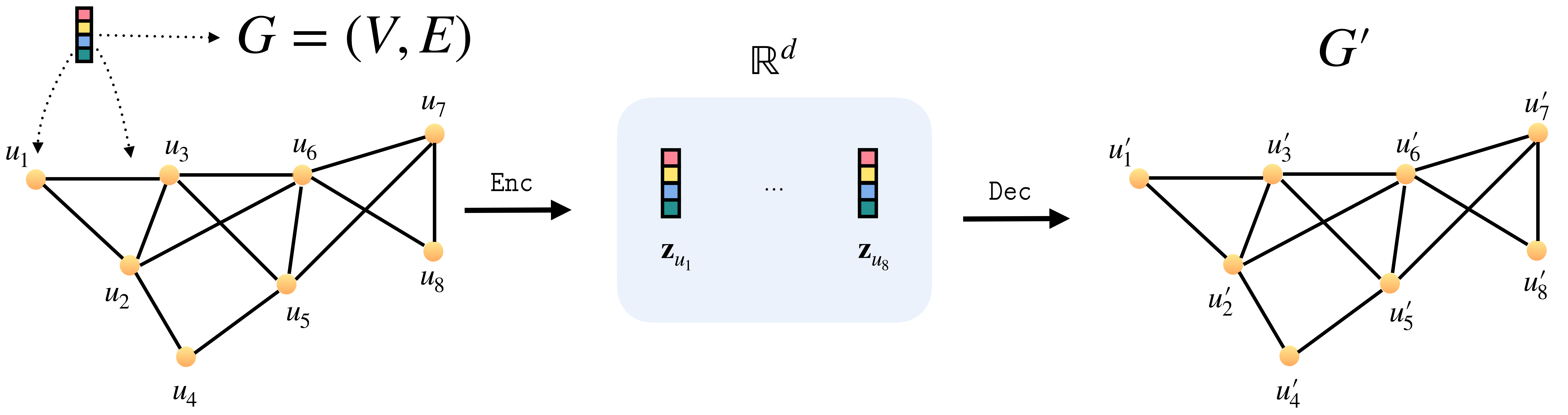
Encoder-Decoder



...and on popular instances of MPNNs

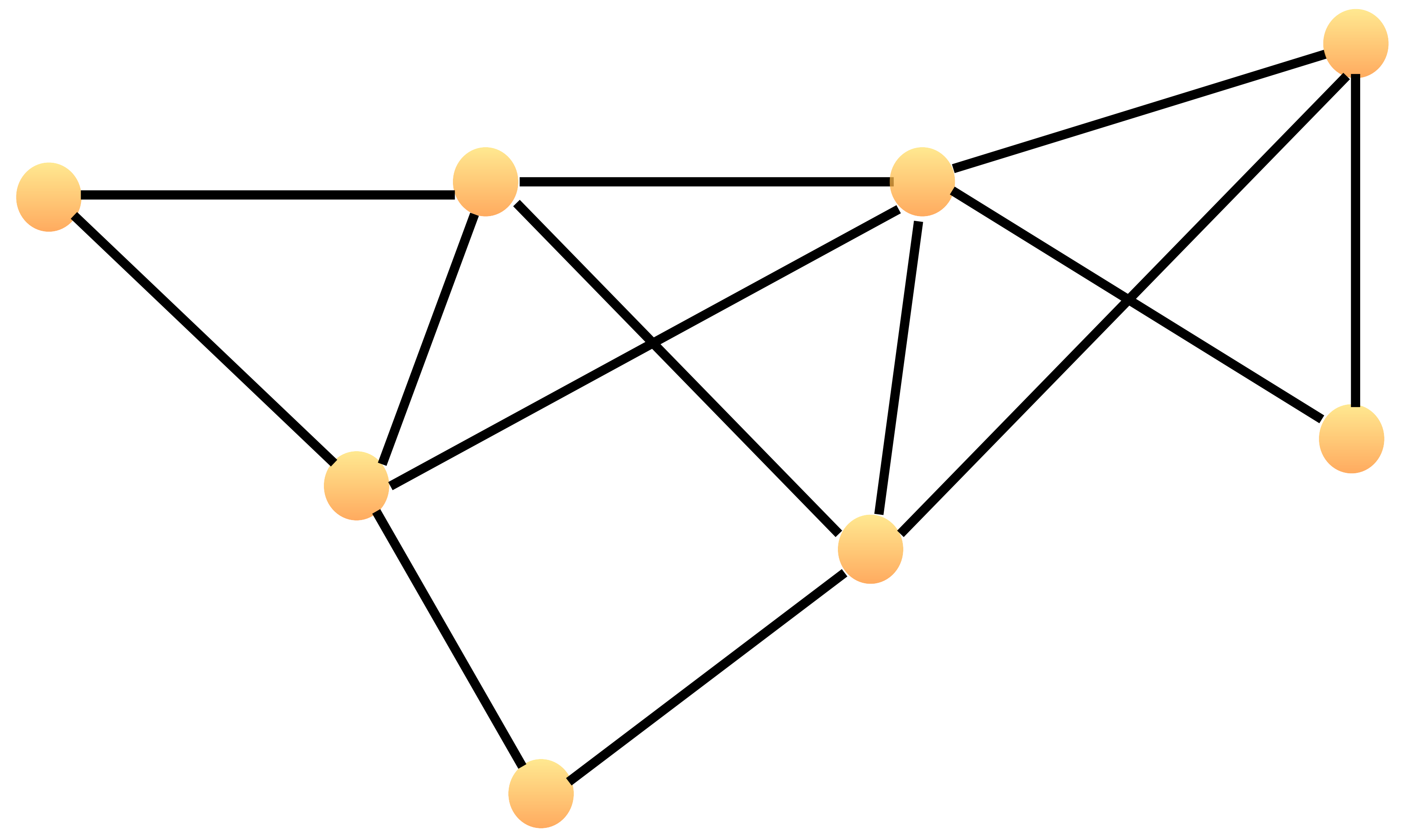
$$\mathbf{h}_u^{(t)} = \sigma\left(\mathbf{W}^{(t)} \sum_{v \in N(u) \cup \{u\}} \alpha_{(u,v)} \, \mathbf{h}_v^{(t-1)}\right)$$

Encoder-Decoder



Today's Lecture: Graph neural networks beyond MPNNs

Graph Neural Networks: A General Perspective



Initial motivation (Lecture 3): Learn functions over graphs with invariance (resp., equivariance) to node orderings: no need to be via a specific message passing framework, or even, message passing.

A more general definition: In a graph neural network, nodes of the input graph are assigned vector representations, which are updated iteratively through series of invariant or equivariant computational layers.

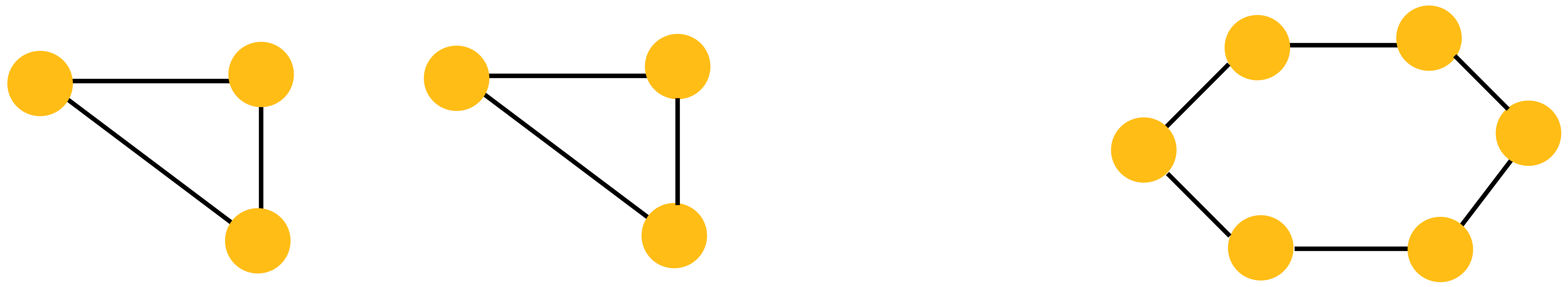
Today's Lecture: Higher-order graph neural networks, which use higher-order representations of the graphs, e.g., higher-order tensors, to be able to approximate a larger class of functions.

Overview

- The Weisfeiler-Lehman hierarchy
- Higher-order graph neural networks
  - Higher-order message passing neural networks: k-GNNs
  - Invariant/Equivariant graph networks
  - Provably powerful graph networks
- Expressive power in real-world data
- Homophily and heterophily
- Summary

The Weisfeiler-Lehman Hierarchy

A Tale of Two Graphs



Refresher: 1-WL cannot distinguish the nodes in the respective graphs — and so neither can MPNNs.

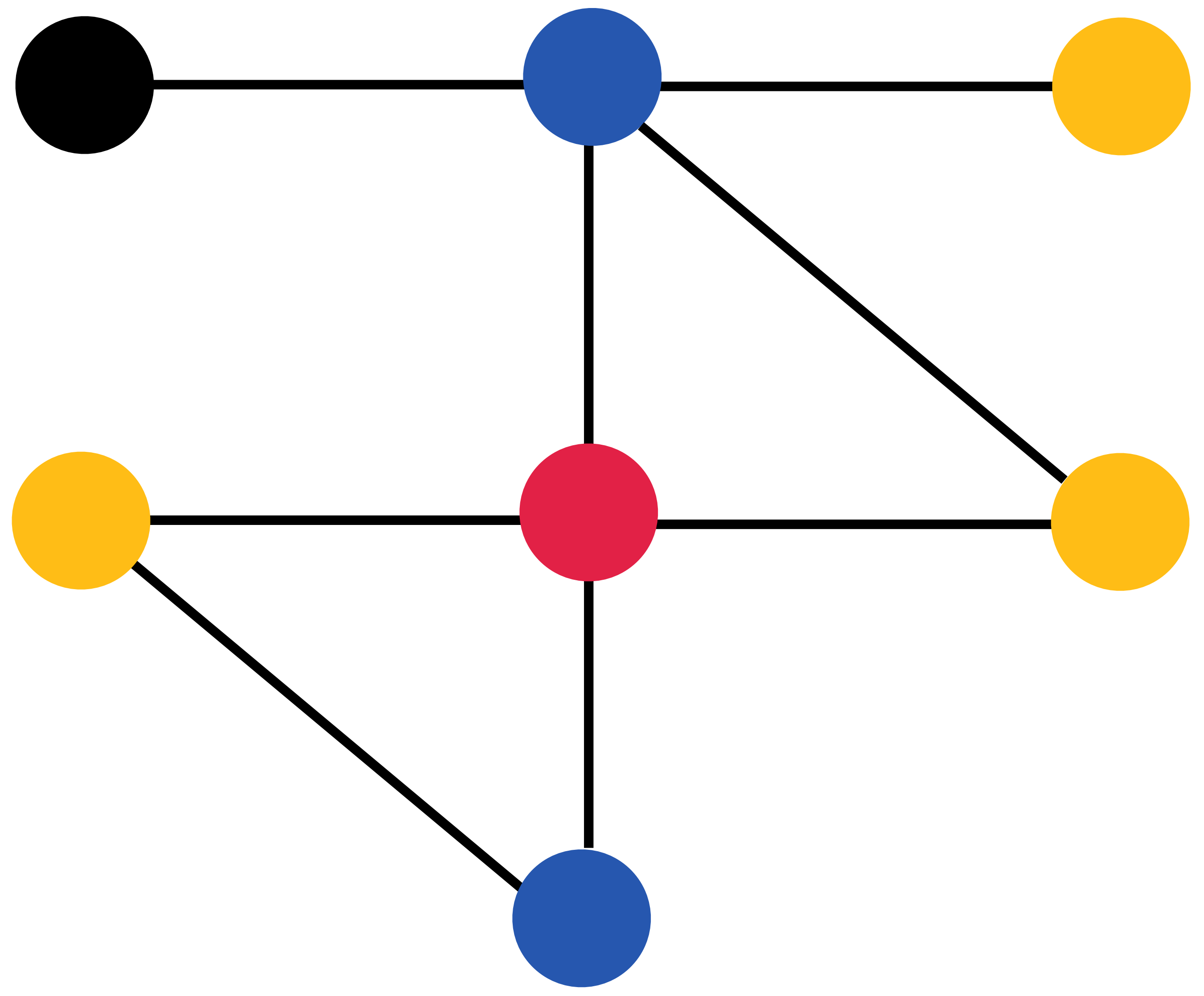
Question: What if we extend the 1-WL algorithm to consider, e.g., pairs of nodes when coloring?

This extended algorithm is called the 2-dimensional WL algorithm, and it can distinguish these two graphs!

Folklore k-dimensional Weisfeiler-Lehman

For a WL-dimension $k \in \mathbb{N}$, the k-WL algorithm is as follows:

- Consider k-tuples $(v_1, \ldots, v_k) \in V_G^k$ of nodes.
- Consider a coloring function $\lambda : V_G^k \mapsto \mathbf{C}$ that colors each k-tuple of nodes of the graph with a color from a set $\mathbf{C}$ of colors.
- This color will depend on the isomorphism type of the tuple, e.g., a k-cycle and a k-tree will have different colors.



Partitions $\pi(\lambda)$ of V_G and the refinement relation $\preceq$ is as before.

Note: A k-tuple is denoted as $t = (u_1, \ldots, u_k)$, a substitution as $t[v/i] = (u_1, \ldots, u_{i-1}, v, u_{i+1}, \ldots, u_k)$ and each coloring respects the isomorphism type of a tuple in graph.

Folklore k-dimensional Weisfeiler-Lehman

Algorithm: Given a graph $G = (V, E)$, a dimension $k \geq 1$, and an initial coloring $\lambda^{(0)}$ of k-tuples:

1. **Initialization**: All k-tuples $t \in V_G^k$, are initialized to their initial colors $\lambda^{(0)}(t)$.
2. **Refinement**: The color of a k-tuple $t = (u_1, \ldots, u_k)$ is refined by combining the colors of its neighborhood, which is defined as the set of all k-tuples in which one node differs from t:

   $$\lambda^{(i+1)}(t) = \mathsf{HASH}\Big(\lambda^{(i)}(t), \{\{\big(\lambda^{(i)}(t[v/1]), \ldots, \lambda^{(i)}(t[v/k])\big) \mid v \in V_G\}\}\Big),$$

   where double-braces denote a multiset, and HASH bijectively maps any pair to a unique value in $\mathbf{C}$.
3. **Stop**: Terminates on a stable coloring is reached, at iteration j, where j is the minimal integer satisfying

   $$\forall t, t' \in V_G^k : \lambda^{(j+1)}(t) = \lambda^{(j+1)}(t') \text{ if and only if } \lambda^{(j)}(t) = \lambda^{(j)}(t').$$

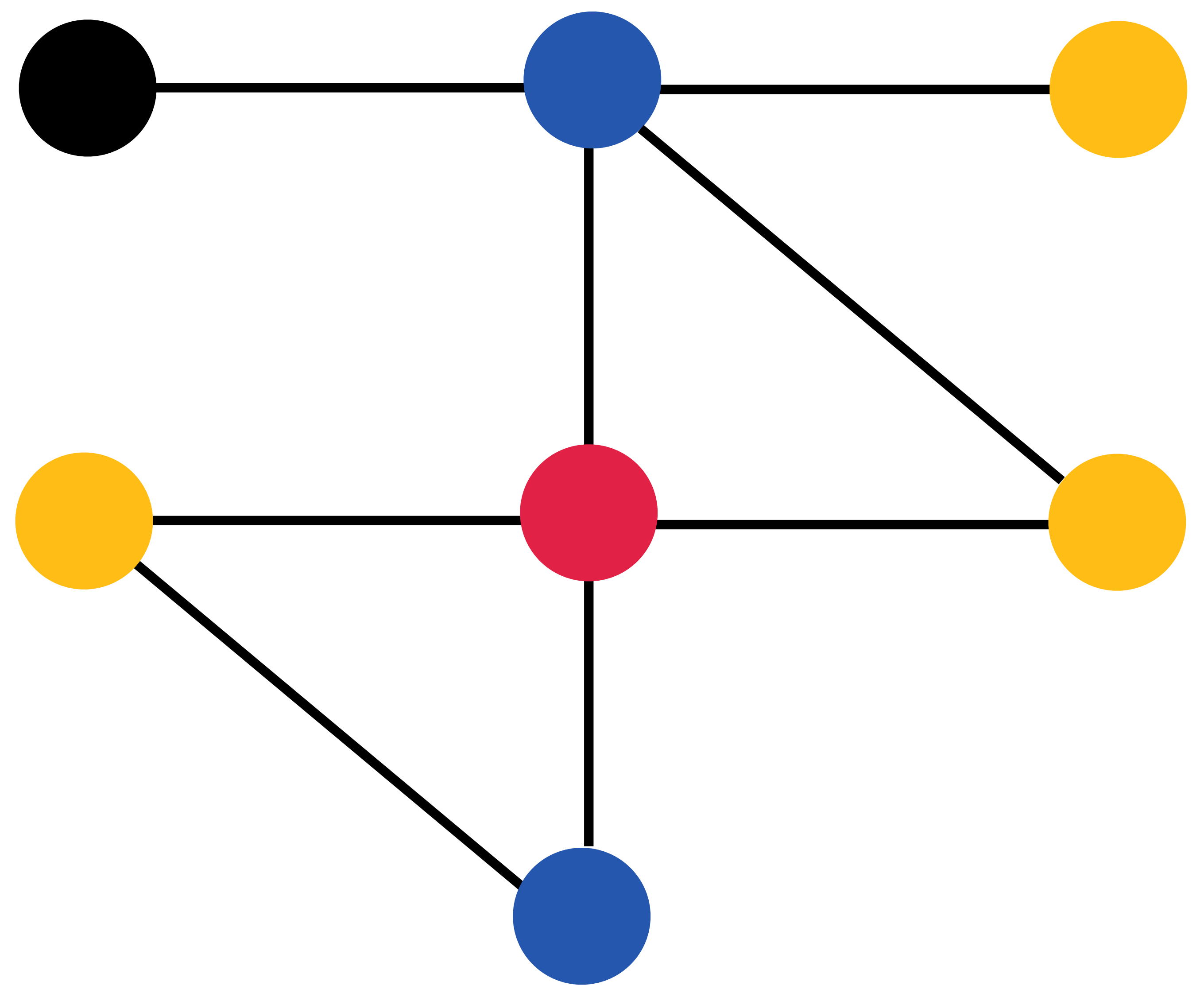
Folklore k-dimensional Weisfeiler-Lehman

k-**WL**: Different versions of k-WL lead to inconsistent dimension counts. We follow Cai et al. (1992), which is also known as the folklore WL algorithm, or k-FWL (Grohe, 2021).

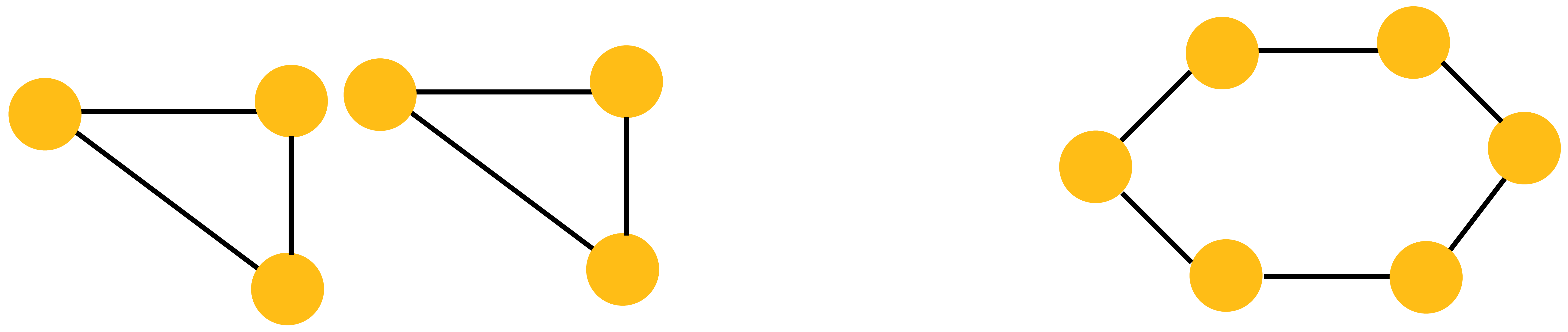
In the non-folklore (oblivious) version, the update step is defined differently based on set of tuples, instead of ordered tuples. Nevertheless:

- Both lead to the same expressive power modulo the shift in k: For any $k \geq 2$, k-FWL is equivalent to $(k+1)$-WL (Grohe, 2017).
- The k-FWL hierarchy is proper: For each $k \geq 1$ there is a pair of non-isomorphic graphs distinguishable by $(k+1)$-FWL but not by k-FWL.
- Non-folklore case: 1-WL and 2-WL have the same expressive power.

We write k-WL to refer to the folklore version, as it is more standard.



A Tale of Two Graphs



Theorem (Cai et al., 1992). For all $k \geq 2$, two graphs G and H satisfy the same C^k-sentences if and only if $(k-1)$-WL does not distinguish them.

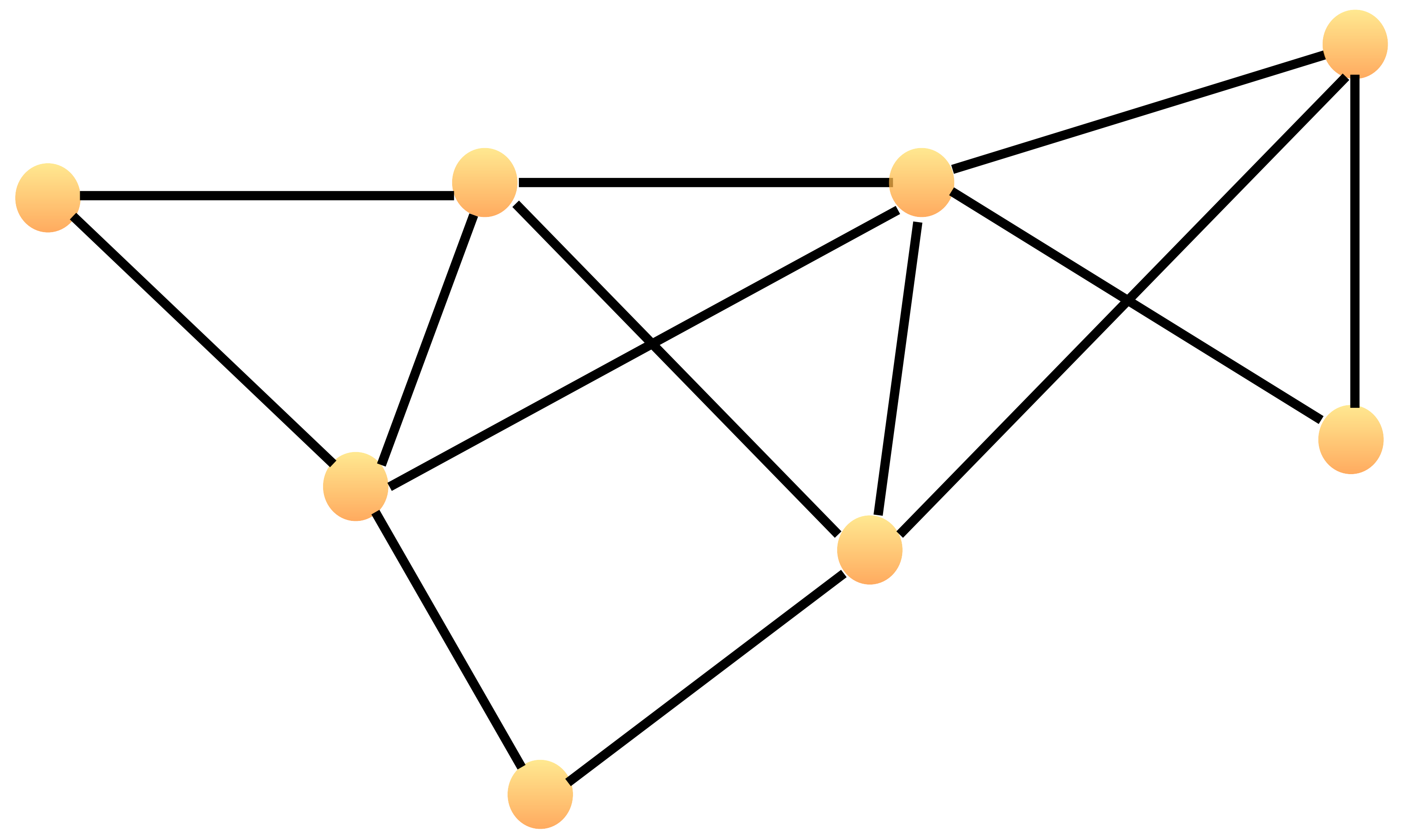
The graphs can be distinguished by the following sentence:

$$\Phi = \exists x, y, z\ E(x,y) \wedge E(y,z) \wedge E(x,z) \wedge (x \neq z) \wedge (x \neq y) \wedge (y \neq z)$$

That is, there are C^3-sentences, distinguishing these graphs, and so must 2-WL.

Higher-Order Graph Neural Networks

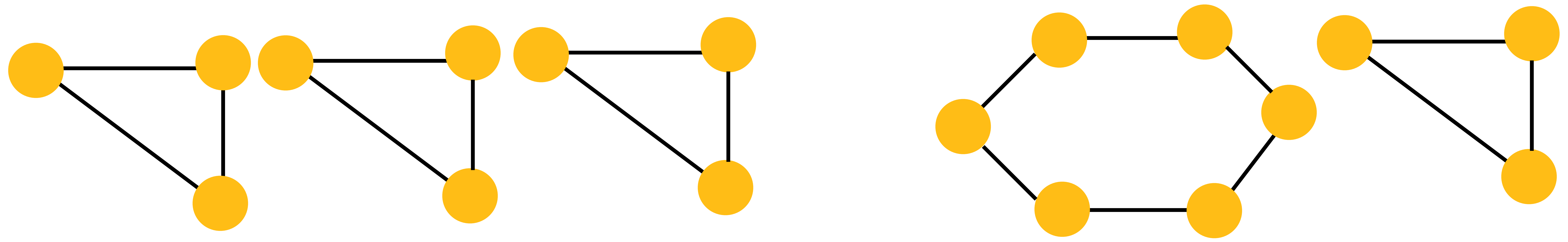
Higher-Order Graph Neural Networks



Higher-order graph neural networks: Graph neural networks which use higher-order representations of the graphs, e.g., higher-order message passing, or higher-order tensors, to be able to approximate a larger class of functions.

Higher-Order Message Passing Neural Networks

Weisfeiler-Lehman: From 1-GNNs to k-GNNs

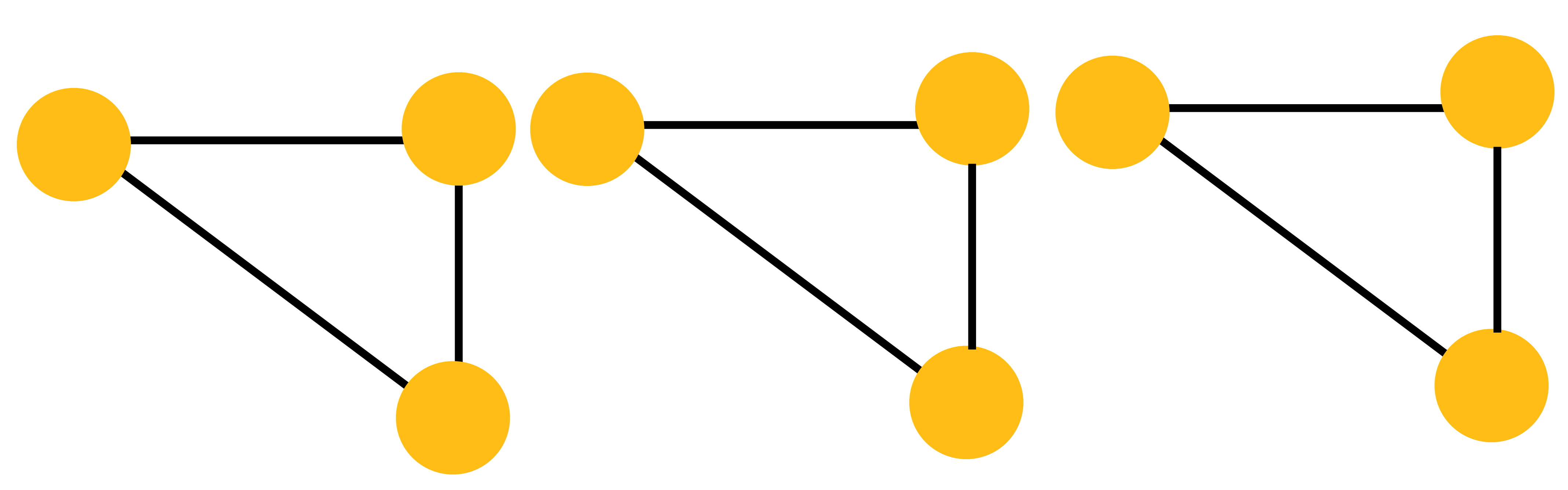


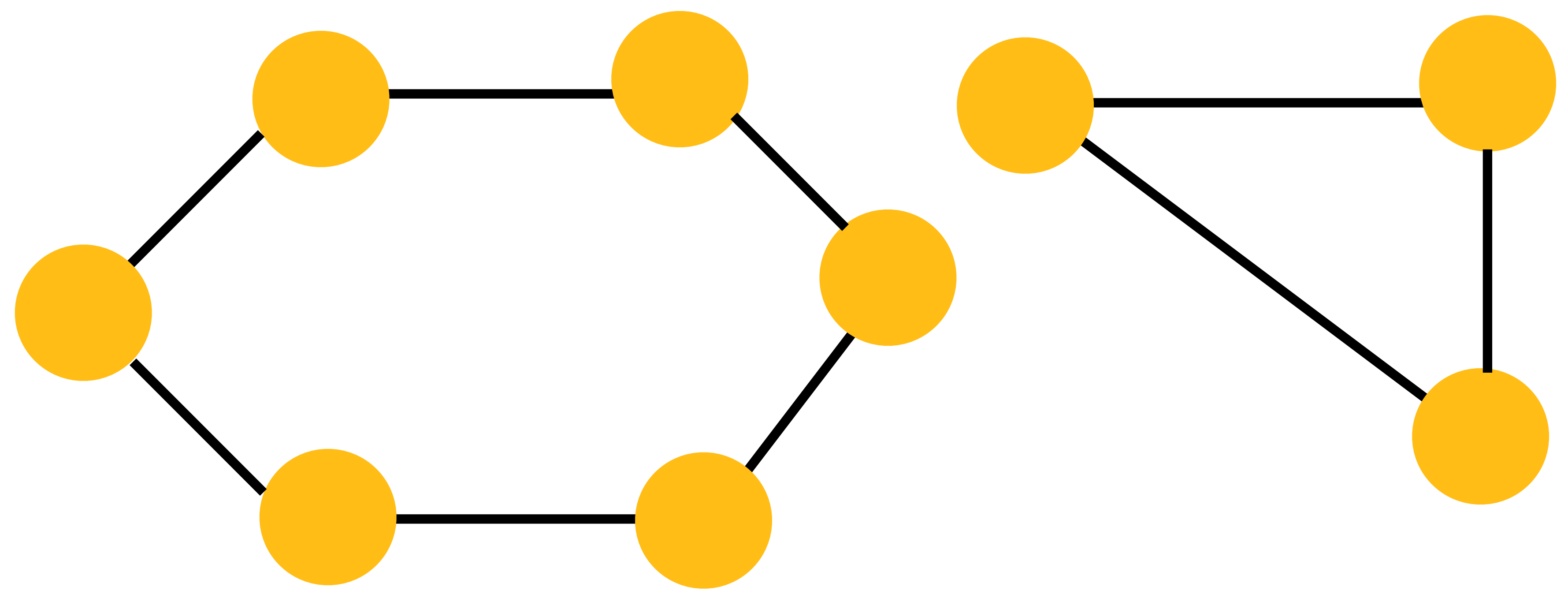
The k-GNN model (Morris et al., 2019) is a generalization of MPNNs based on the $(k-1)$-WL algorithm.

Idea: Higher-order message passing between subgraph structures, rather than individual nodes.

Intuition: This form of message passing can capture structural information that is not visible at the node-level.

Weisfeiler-Lehman: From 1-GNNs to k-GNNs





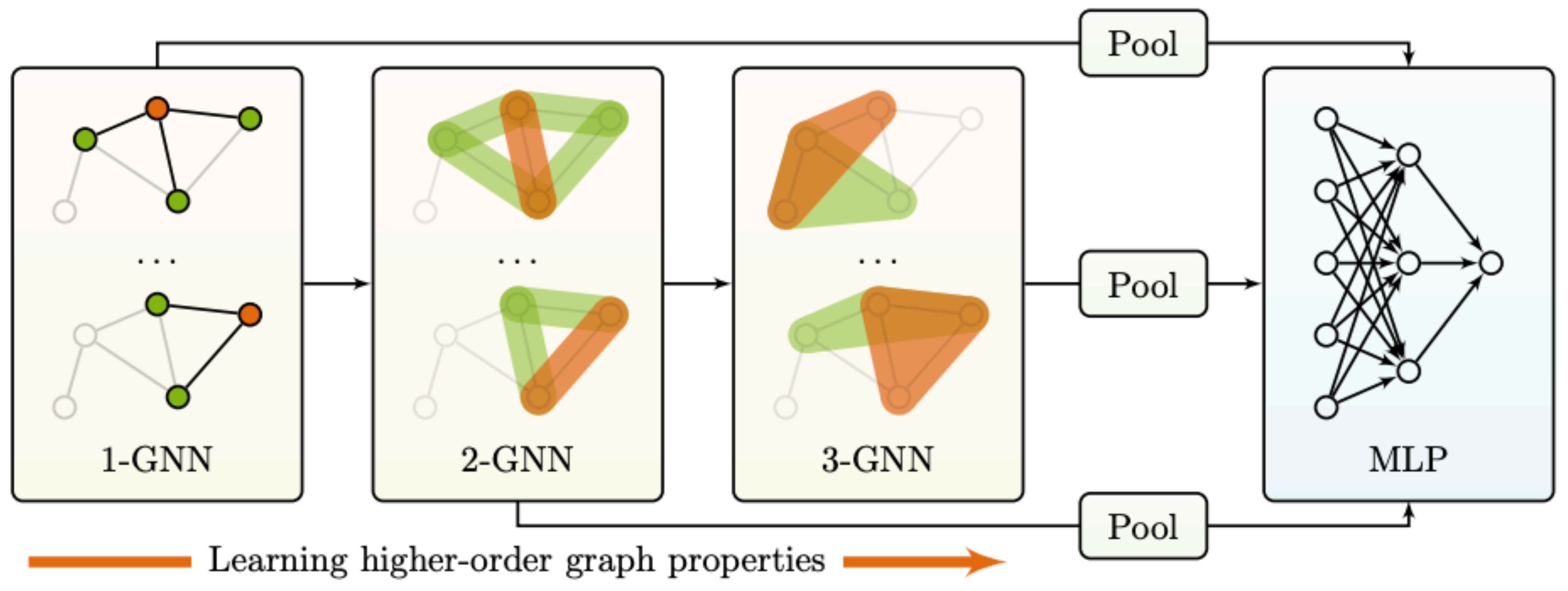
3-GNNs of (Morris et al., 2019) have the same power as folklore 2-WL and can distinguish these graphs.

The C^3 formula characterizes a property that distinguishes these graphs:

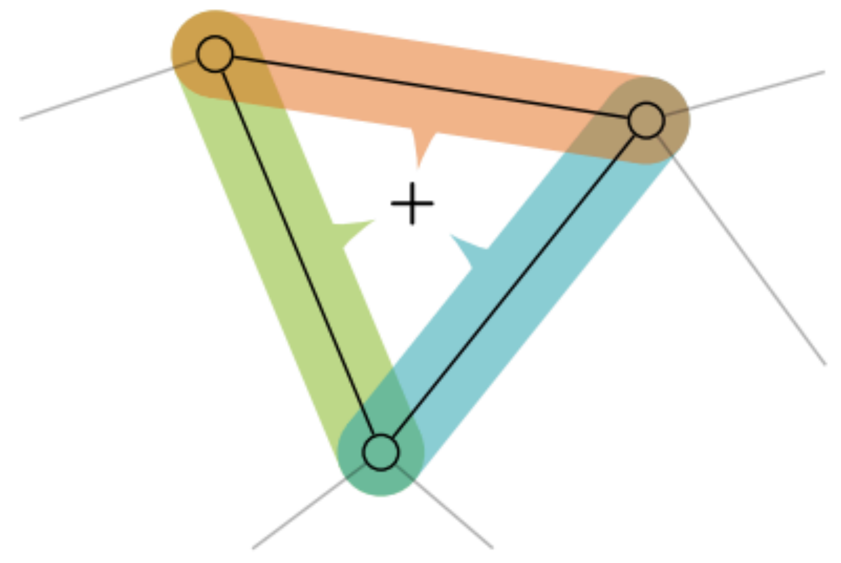
$$\Phi = \exists x, y, z \; E(x, y) \land E(x, z) \land \neg E(y, z) \land (x \neq z) \land (x \neq y) \land (y \neq z)$$

C^3 can distinguish these graphs → 2-WL and hence 3-GNN can distinguish these graphs.

Hierarchical Variants



(a) Hierarchical 1-2-3-GNN network architecture



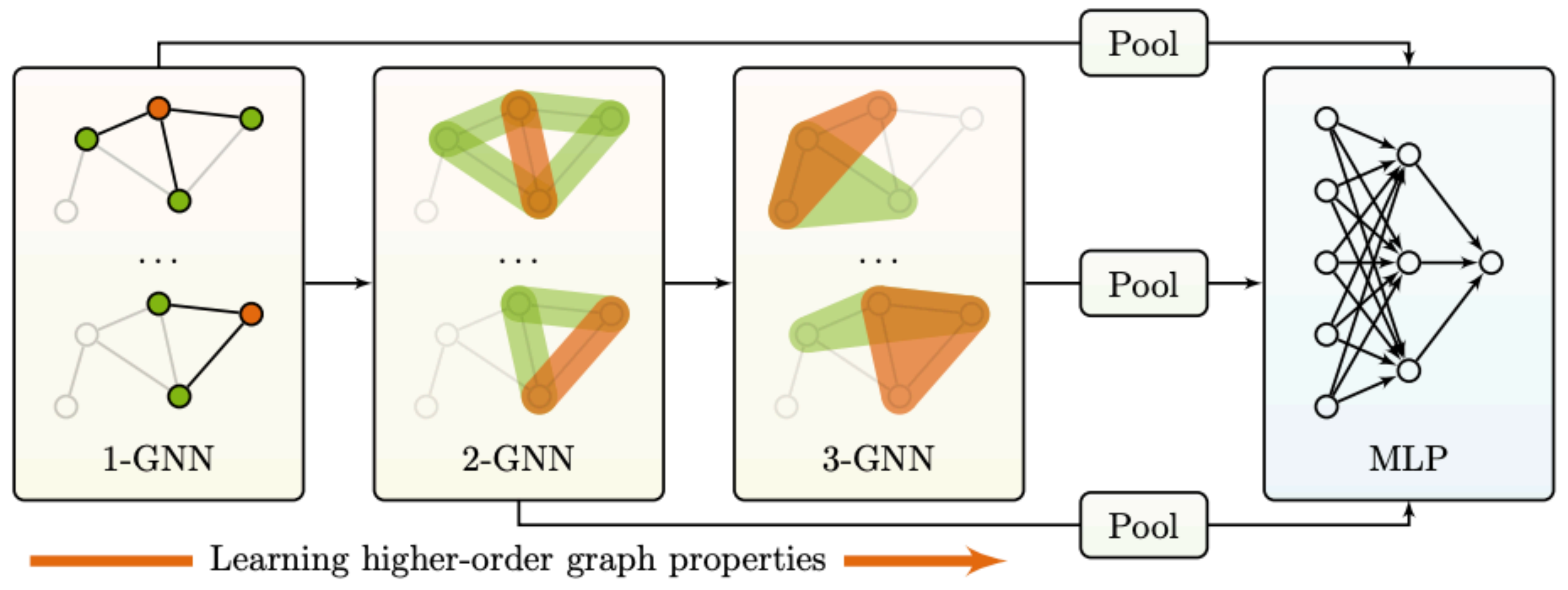
(b) Pooling from 2- to 3-GNN.

Figure 1: Illustration of the proposed hierarchical variant of the k-GNN layer. For each subgraph S on k nodes a feature f is learned, which is initialized with the learned features of all $(k-1)$-element subgraphs of S. Hence, a hierarchical representation of the input graph is learned.

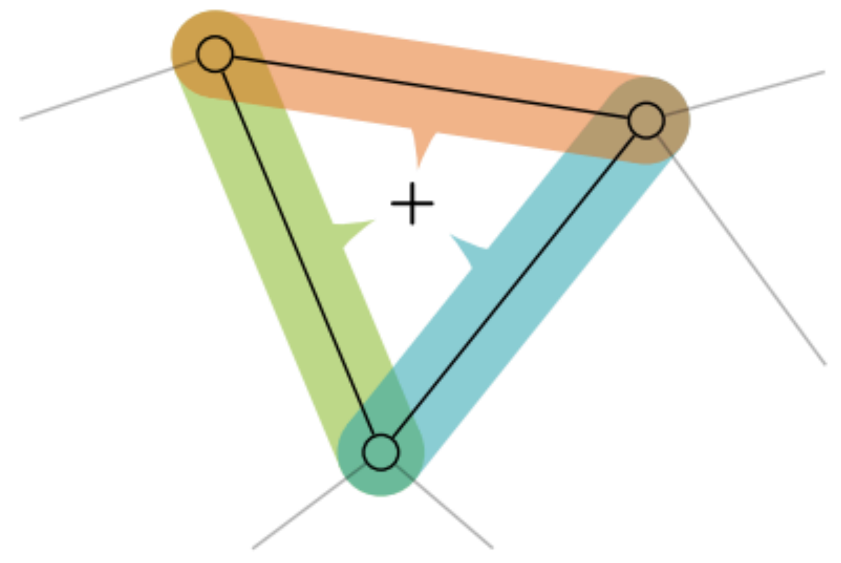
Hierarchical variants of k-GNNs: 1-k-GNNs combine representations learned at different granularities.

Idea: Applying the usual node-level message passing (1-WL), and then using the resulting representations to learn representations for pairs of nodes, with a higher-order message passing (2-WL), etc.

Hierarchical Variants



(a) Hierarchical 1-2-3-GNN network architecture



(b) Pooling from 2- to 3-GNN.

Figure 1: Illustration of the proposed hierarchical variant of the k-GNN layer. For each subgraph S on k nodes a feature f is learned, which is initialized with the learned features of all $(k-1)$-element subgraphs of S. Hence, a hierarchical representation of the input graph is learned.

Intuition: Initial messages in a k-GNN are based on the output of lower-dimensional GNNs, which allows the model to effectively capture graph structures of varying granularity.

Practice: Many real-world graphs inherit a hierarchical structure, and so a hierarchical message passing approach is potentially helpful — and this is empirically confirmed in the evaluation (Morris et al., 2019).

Limitations of k-GNNs

Excessive memory requirements: k-GNNs have $(k-1)$-WL expressive power, but need $O(|V|^k)$ memory to run. These higher-order models require intractably-sized intermediate tensors in practice.

In fact, it is implemented only up to 3-GNNs (corresponding to 2-WL expressiveness), which already requires cubic memory allocations — already intractable on existing benchmarks.

Time complexity: The complexity message passing also increases combinatorially in k!

Power: k-GNNs are more expressive than MPNNs, but still limited in their expressive power, as $(k+1)$-GNN is strictly more expressive than k-GNN for any $k \geq 2$.

Inductive bias: Though permutation-invariant, the non-hierarchical version of the algorithm can somewhat lose the explicit connection to node-level information, as only k-tuples are considered.

Invariant/Equivariant Graph Networks

Invariant/Equivariant Graph Networks

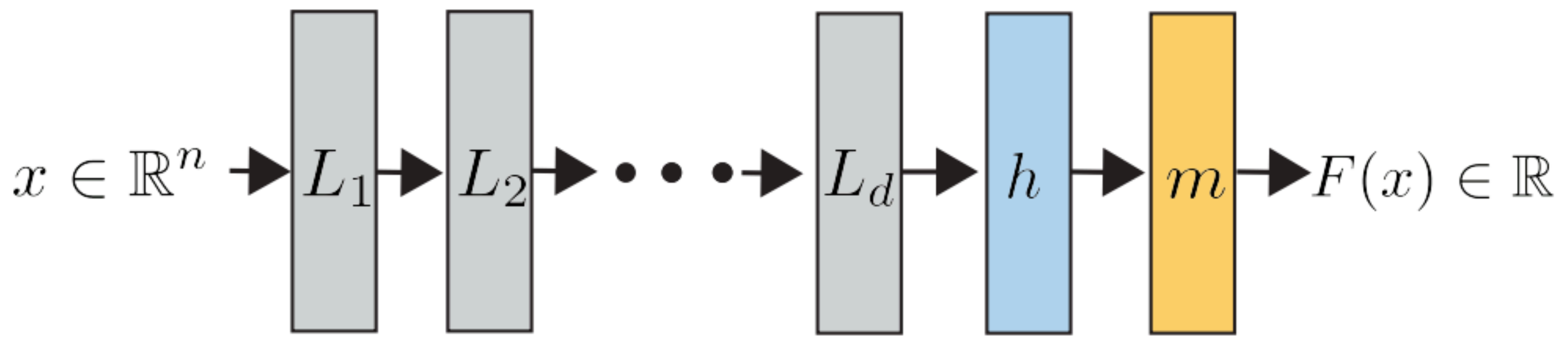


Figure 1. Illustration of invariant network architecture. The function is composed of multiple linear G-equivariant layers (gray), possibly of high order, and ends with a linear G-invariant function (light blue) followed by a Multi Layer Perceptron (yellow).

Idea: A GNN model based on permutation equivariant/invariant tensor operations.

Input: A tensor $\mathbf{X} \in \mathbb{R}^{|V|^2 \times d}$, where the first two channels correspond to the adjacency matrix of the graph and the remaining channels encode the initial node features.

Invariant/Equivariant Graph Networks

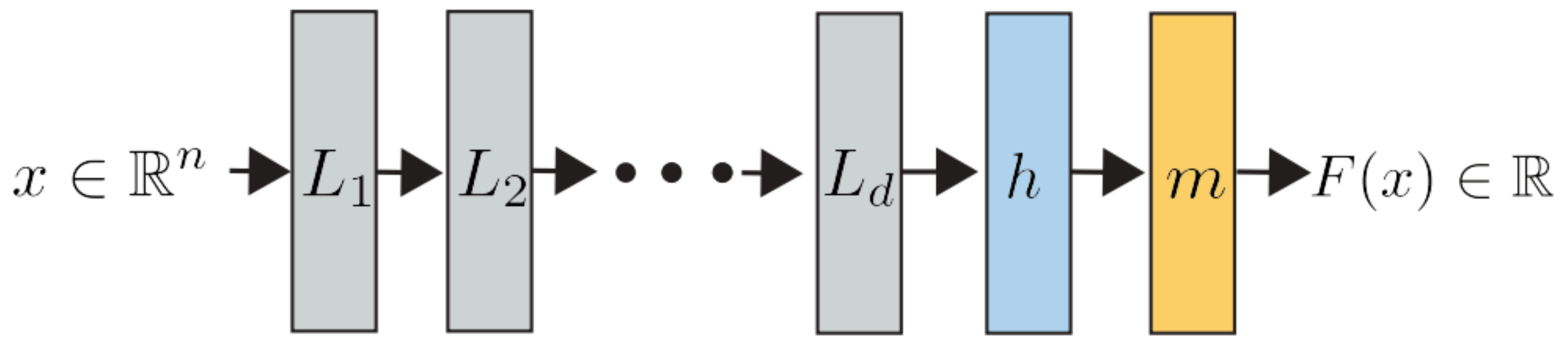


Figure 1. Illustration of invariant network architecture. The function is composed of multiple linear G-equivariant layers (gray), possibly of high order, and ends with a linear G-invariant function (light blue) followed by a Multi Layer Perceptron (yellow).

A linear invariant layer can be defined as $\mathscr{L} : \mathbb{R}^{|V|^k \times d_1} \mapsto \mathbb{R}^{d_2}$ such that for all permutations π:

$$\mathscr{L}(\pi(\mathbf{X})) = \mathscr{L}(\mathbf{X}).$$

Invariant/Equivariant Graph Networks

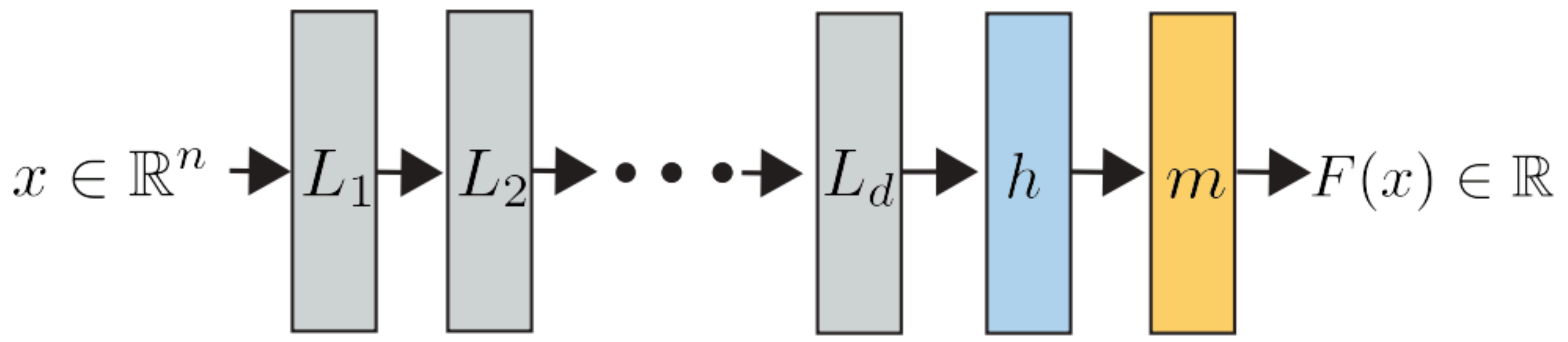


Figure 1. Illustration of invariant network architecture. The function is composed of multiple linear G-equivariant layers (gray), possibly of high order, and ends with a linear G-invariant function (light blue) followed by a Multi Layer Perceptron (yellow).

A linear equivariant layer can be defined as $\mathscr{L} : \mathbb{R}^{|V|^{k_1} \times d_1} \mapsto \mathbb{R}^{|V|^{k_2} \times d_2}$ such that for all permutations π:

$$\mathscr{L}(\pi(\mathbf{X})) = \pi(\mathscr{L}(\mathbf{X})).$$

Invariant/Equivariant Graph Networks

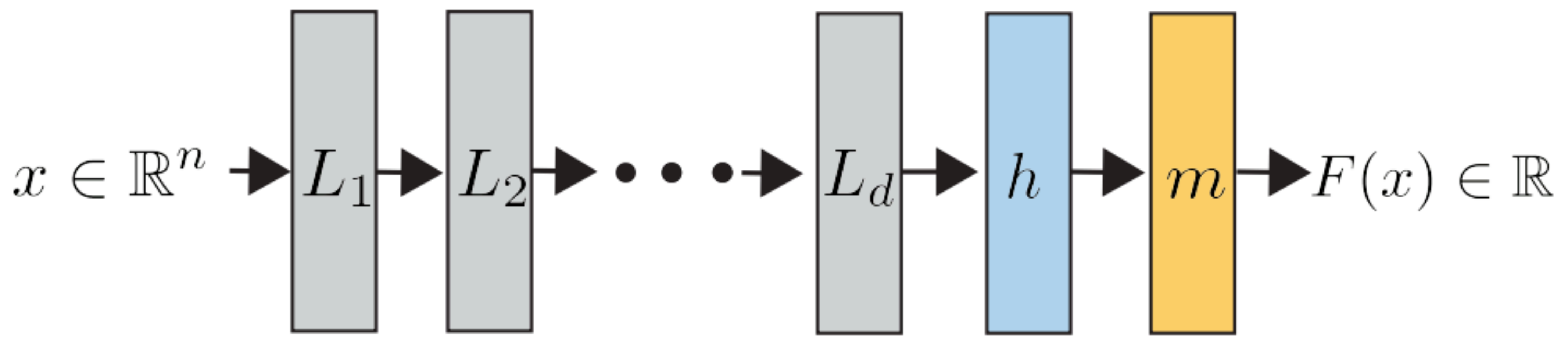


Figure 1. Illustration of invariant network architecture. The function is composed of multiple linear G-equivariant layers (gray), possibly of high order, and ends with a linear G-invariant function (light blue) followed by a Multi Layer Perceptron (yellow).

Intermediate representations $\mathbf{X} \in \mathbb{R}^{|V|^k \times d}$: k-order tensors where the first k channels are indexed by the nodes of the graph.

Higher-order: The model is called k-order, as it allows invariant/equivariant layers with k channels, and this directly correlates with the expressive power of the model.

Remark: The hidden variables can be tensors of arbitrary order - even if the input tensor is low-order.

Invariant/Equivariant Graph Networks

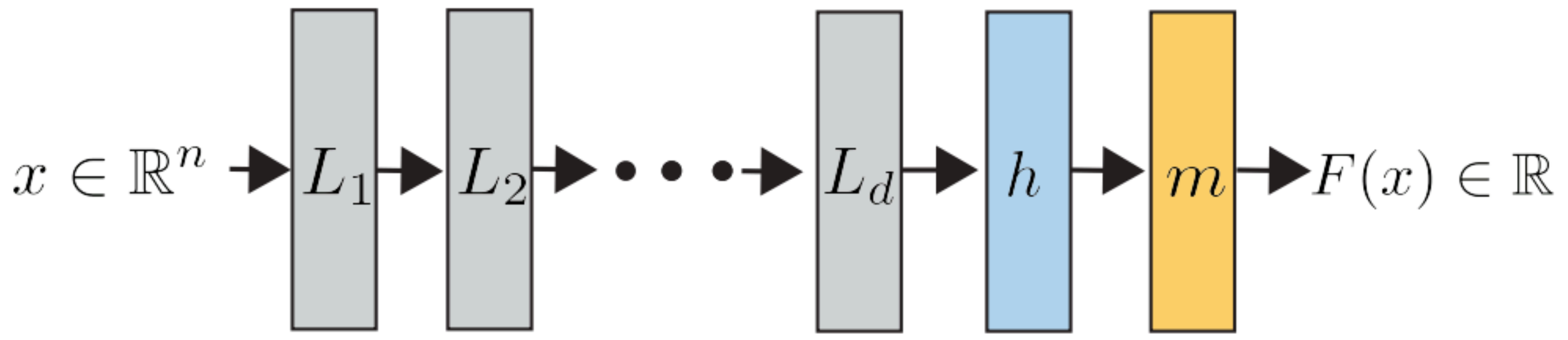


Figure 1. Illustration of invariant network architecture. The function is composed of multiple linear G-equivariant layers (gray), possibly of high order, and ends with a linear G-invariant function (light blue) followed by a Multi Layer Perceptron (yellow).

Invariant k-order GNNs (Maron et al., 2019b), or k-IGNs, is defined as:

$$F = \mathrm{MLP} \circ \mathscr{H} \circ \mathscr{L}_d \circ \sigma \circ \cdots \circ \sigma \circ \mathscr{L}_1,$$

where $\mathscr{L}_1, \ldots, \mathscr{L}_d$ are equivariant linear layers (with up to k different channels), $\mathscr{H}$ is an invariant layer, and σ denotes element-wise non-linearity.

Invariant/Equivariant Graph Networks

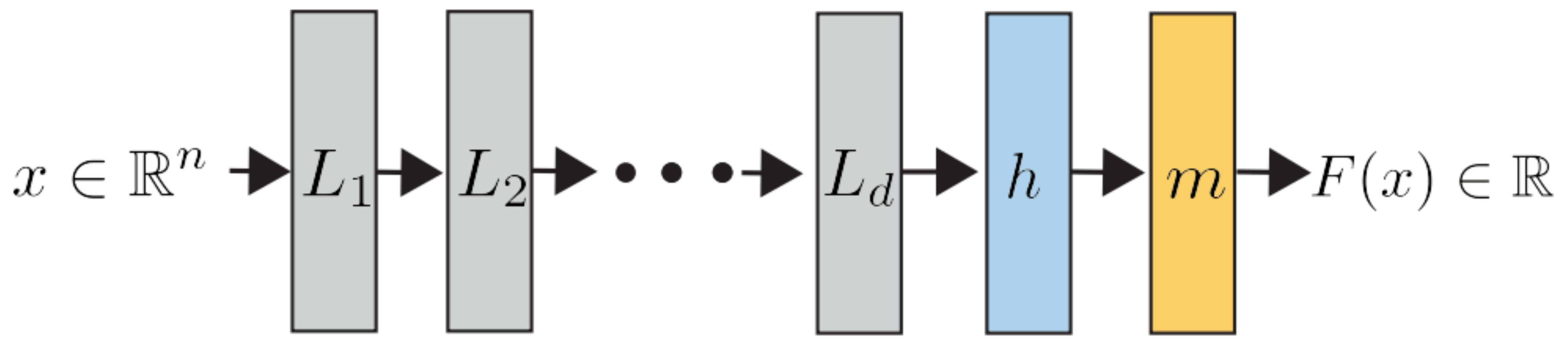


Figure 1. Illustration of invariant network architecture. The function is composed of multiple linear G-equivariant layers (gray), possibly of high order, and ends with a linear G-invariant function (light blue) followed by a Multi Layer Perceptron (yellow).

Invariant by construction:

$$\begin{aligned} F(\pi(\mathbf{X})) &= \mathrm{MLP}(\mathscr{H}(\mathscr{L}_d(\cdots(\mathscr{L}_1(\pi(\mathbf{X})))\cdots) \\ &= \mathrm{MLP}(\mathscr{H}(\mathscr{L}_d(\cdots(\pi(\mathscr{L}_1(\mathbf{X})))\cdots) \\ &= \mathrm{MLP}(\mathscr{H}(\pi(\mathscr{L}_d(\cdots((\mathscr{L}_1(\mathbf{X})))\cdots) = F(\mathbf{X}) \end{aligned}$$

Invariant/Equivariant Graph Networks

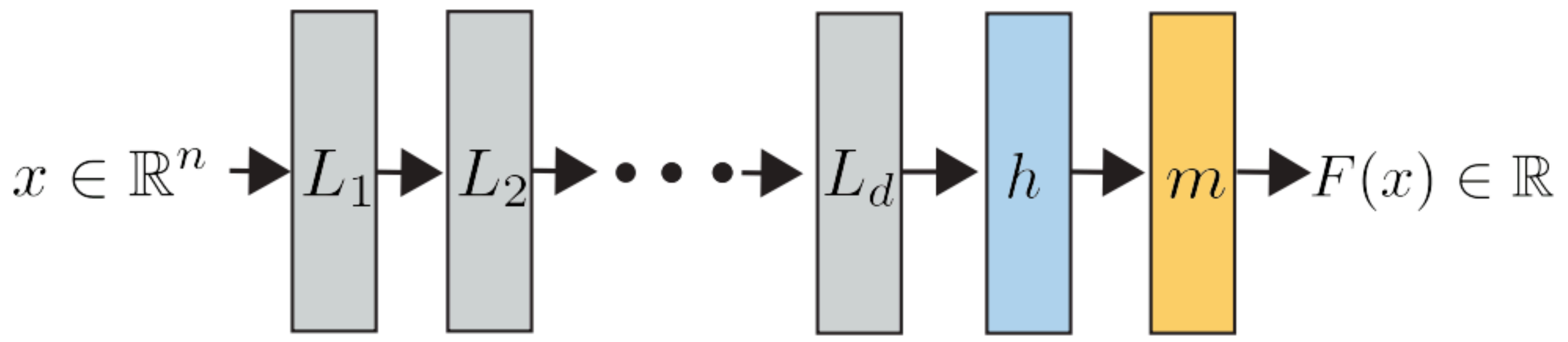


Figure 1. Illustration of invariant network architecture. The function is composed of multiple linear G-equivariant layers (gray), possibly of high order, and ends with a linear G-invariant function (light blue) followed by a Multi Layer Perceptron (yellow).

Characterizing linear layers: Maron et al. (2019b) characterize all invariant (resp., equivariant) linear functions, showing that they live in vector spaces of dimension $b(k)$ (respectively, $b(k+l)$), where $b(i)$ is the i-th Bell number.

Parametrization: The dimension of this space does not depend on $|V|$, but only on the order of the input and output tensors - parameterize linearly for all $|V|$ such an operator by the same set of coefficients.

Expressive Power of Invariant Graph Networks

Expressive power: k-IGNs are as powerful as $(k-1)$-WL test.

Theorem 1 (Maron et al., 2019a). For $k > 1$, any graphs G, G' that can be distinguished by the $(k-1)$-WL graph isomorphism test, there exists a k-order network F so that $F(G) \neq F(G')$. On the other direction for every two isomorphic graphs G, G' and k-order network F, $F(G) = F(G')$.

Remark: We are using the folklore variant of WL and the original theorem refers to the oblivious version.

If we bound the size of the input graphs with n, then n-th order invariant networks can distinguish any pair of non-isomorphic graphs. Invariant networks with order-2 tensors could already be computationally challenging!

Universality (Maron et al., 2019c): IGNs are universal, but with tensor order of $\dfrac{n(n-1)}{2}$.

An alternative proof is given by (Keriven and Peyré, 2019), who also showed universality result for EGNs.

Limitations of Invariant Graph Networks

Excessive memory and time requirement: Prohibitive to run for large values of k, due to their very large memory and computational requirements.

Power: k-IGNs are more expressive than MPNNs, and even universal, but with high-order tensors.

Inductive bias: Similarly to k-GNNs, k-IGNs may lose the inductive bias of node information relative to standard MPNNs.

Representations: The correspondence with node-level representations and interactions are implicit. This is unlike k-GNNs, where tuples have representations that are explicitly maintained and updated.

Information propagation: Not solely through edge-connected nodes. Indeed, k-IGNs are inherently designed for graph-level computations which are more global.

Provably Powerful Graph Networks

Provably Powerful Graph Networks

Provably powerful graph networks (PPGNs) are special type of invariant networks, motivated by the search for more expressive, yet still scalable, GNN models:

$$F = \mathrm{MLP} \circ \mathscr{H} \circ \mathscr{B}_d \circ \cdots \circ \mathscr{B}_1,$$

where, $\mathscr{H}$ is an invariant layer, and $\mathscr{B}_1, \ldots, \mathscr{B}_d$ are blocks have the structure shown in Figure 2 of (Maron et al., 2019a).

Idea: Given an input $\mathbf{X} \in \mathbb{R}^{|V| \times |V| \times d}$ apply MLPs in each block to each feature of the input tensor independently (i.e., 3 MLPs), and then perform matrix multiplication between matching features.

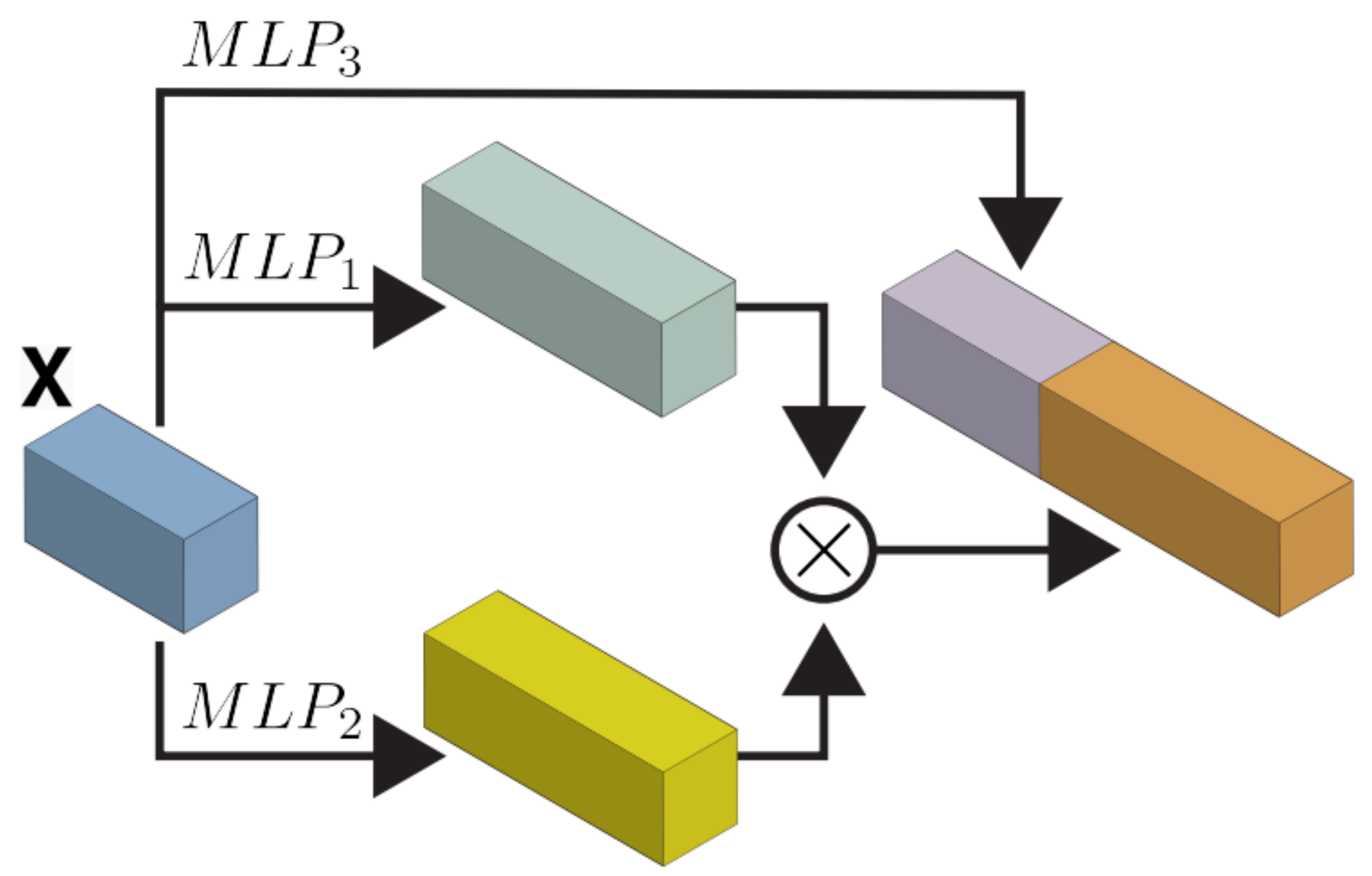


Figure 2: Block structure.

Provably Powerful Graph Networks

Invariant: Matrix multiplication is equivariant, and so the building block is equivariant, which makes the overall function invariant.

Expressive power: PPGNs are strictly more powerful than MPNNs. In fact, PPGNs can distinguish any pair of graphs that can be distinguished by 2-WL.

Intuitively, the matrix multiplication yields a richer aggregation, which enables 2-WL aggregation.

Memory requirements: PPGNs have the same power as 3-GNNs, but they maintain only $O(n^2)$ embeddings, which makes them more memory-efficient than 3-GNNs.

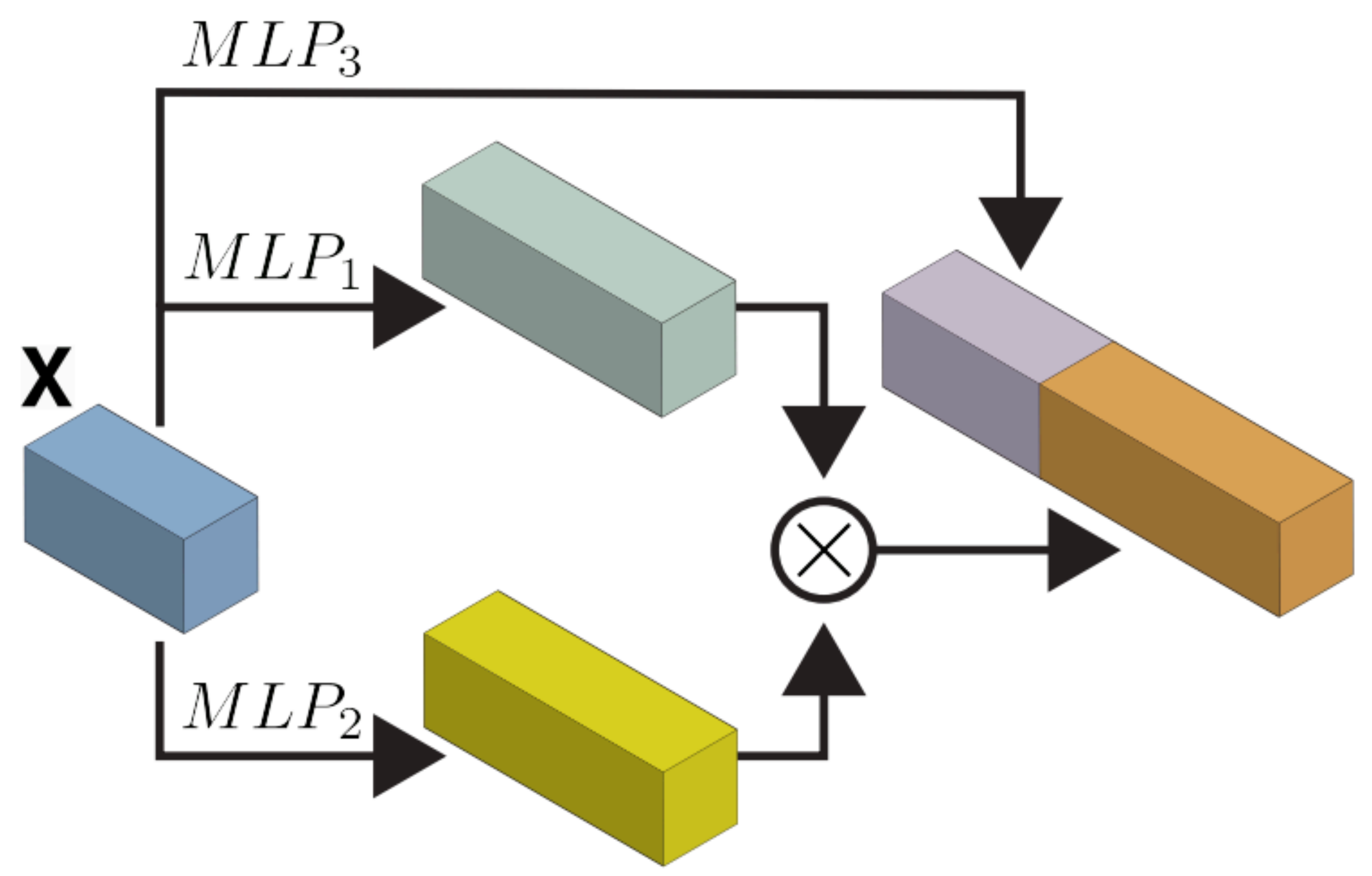


Figure 2: Block structure.

Expressive Power in the Real World

Expressive Power in Real-World Data

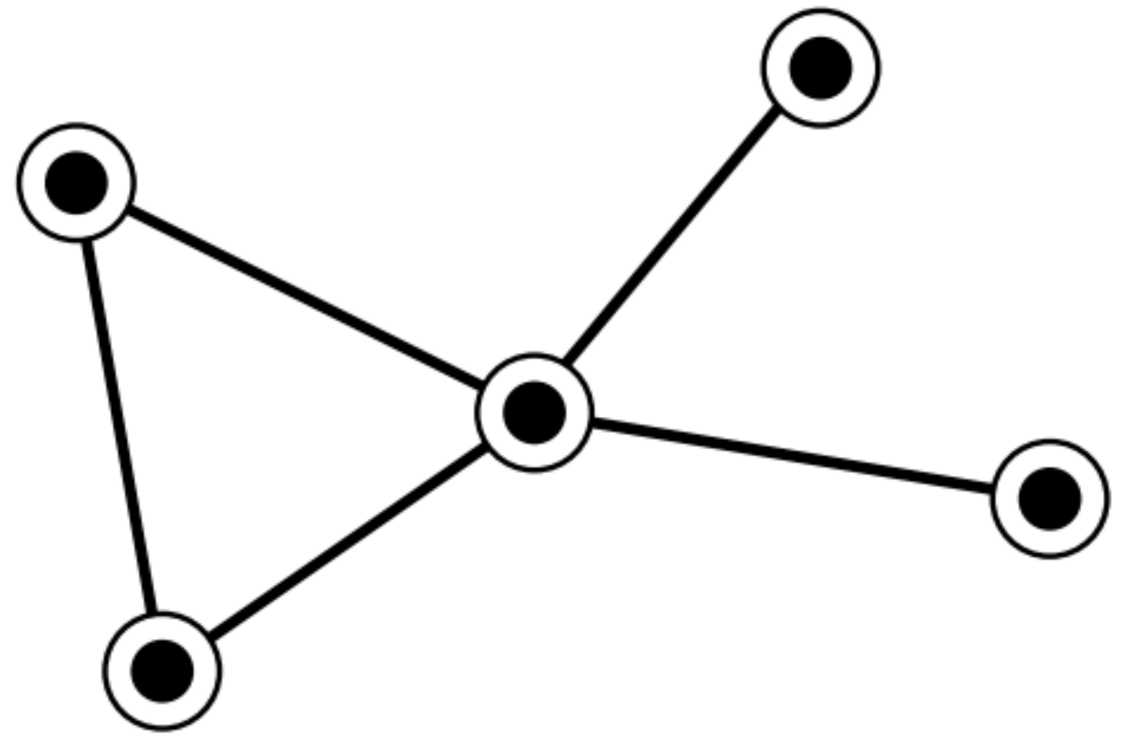


Figure 5 of (Newman, 2013)

MPNNs cannot distinguish very basic graph pairs, but this limitation is not very pronounced empirically, as modern-day benchmarks are unlikely to include limiting cases.

Variability: 1-WL edge cases typically correspond to data that is highly regular, whereas real-world data is overwhelmingly uneven and variable.

Size: Real-world graphs are typically large, and involve thousands, and potentially millions, of nodes: 1-WL can distinguish almost all graphs as the number of graph nodes tends to infinity (Babai et al., 1980).

Expressive Power in Real-World Data

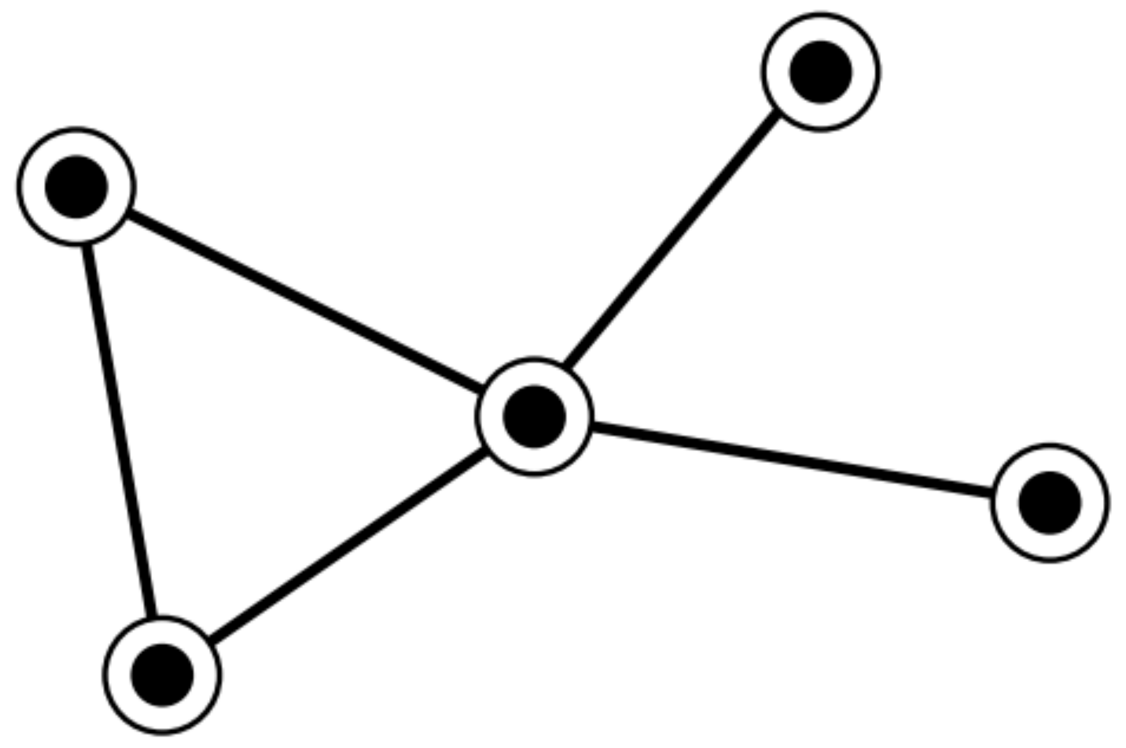


Figure 5 of (Newman, 2013)

MPNNs cannot distinguish very basic graph pairs, but this limitation is not very pronounced empirically, as modern-day benchmarks are unlikely to include limiting cases.

Node features: Rich features on most real-world graphs, yielding unique node features: 1-WL can distinguish all such graphs! Positional encodings to enrich the node features, or randomized features: Lecture 7.

Datasets: Synthetic datasets dedicated to quantify the effect of expressive power are proposed (Abboud et al., 2021) with a detailed comparison against higher-order models: Lecture 7.

Expressive Power in Real-World Data

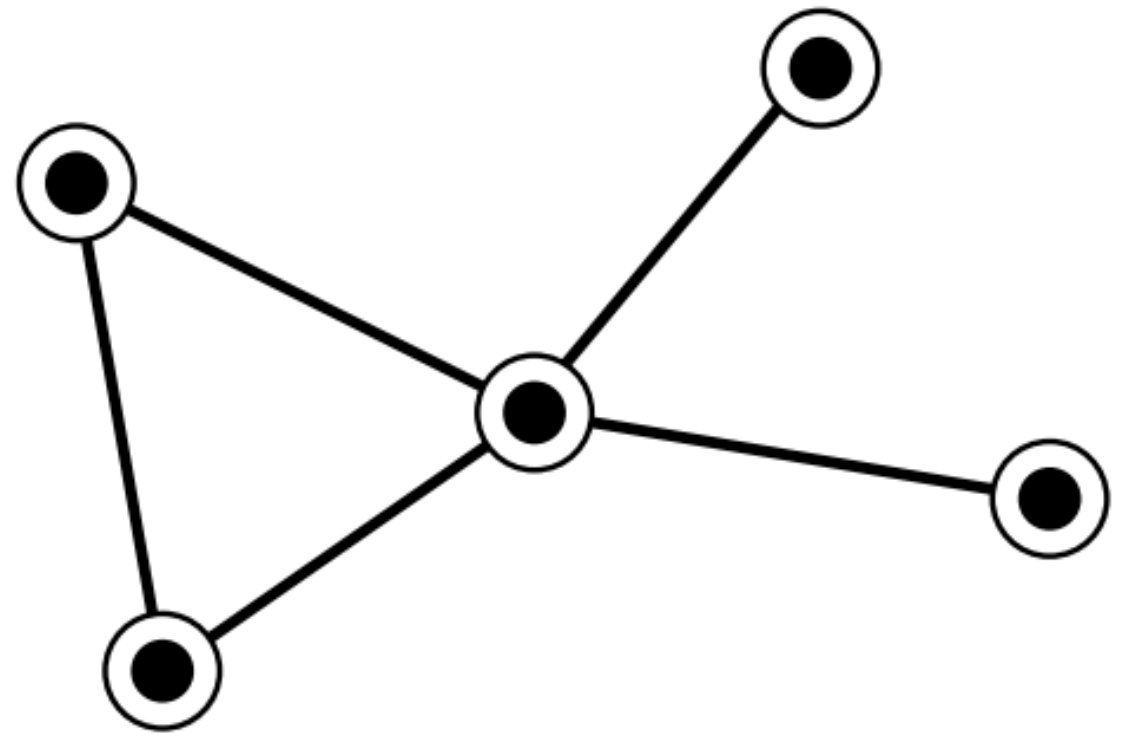


Figure 5 of (Newman, 2013)

There is an excellent survey covering types of graphs observed in real-world data (Newman, 2013):

"In many networks it is found that if vertex A is connected to vertex B and vertex B to vertex C, then there is a heightened probability that vertex A will also be connected to vertex C. In the language of social networks, the friend of your friend is likely also to be your friend."

$$C = 3 \times \frac{\#\text{triangles in the network}}{\#\text{connected triples of vertices}}$$

Expressive Power in Real-World Data

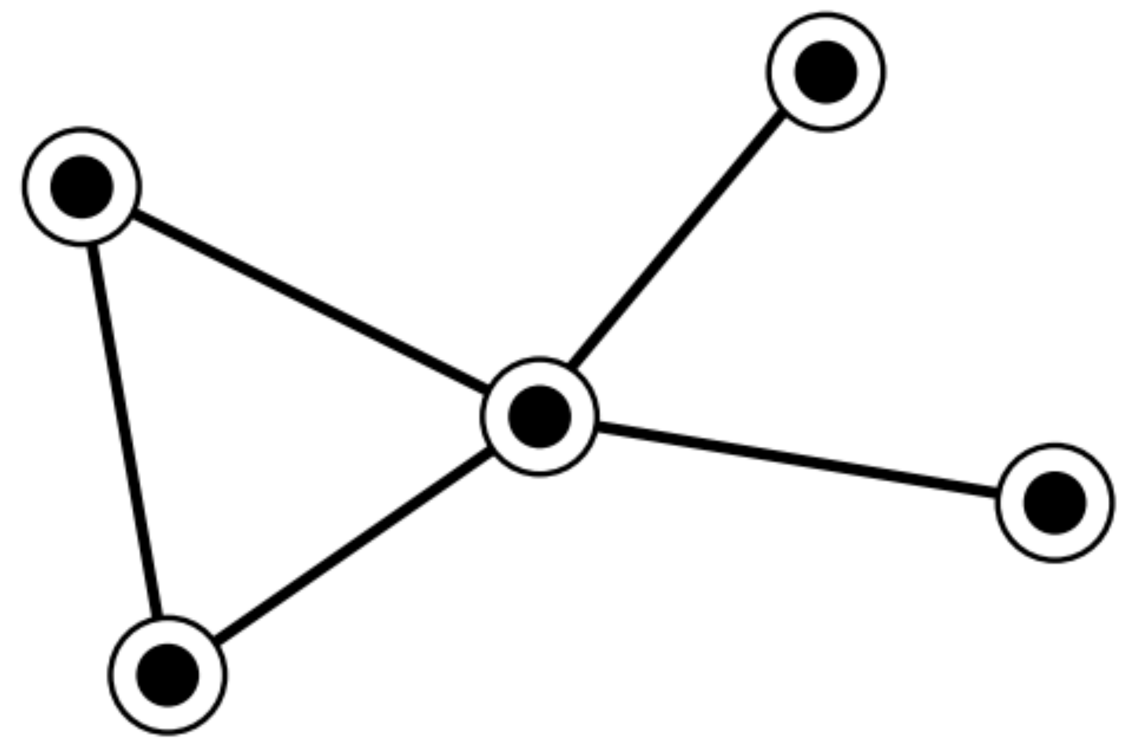


Figure 5 of (Newman, 2013)

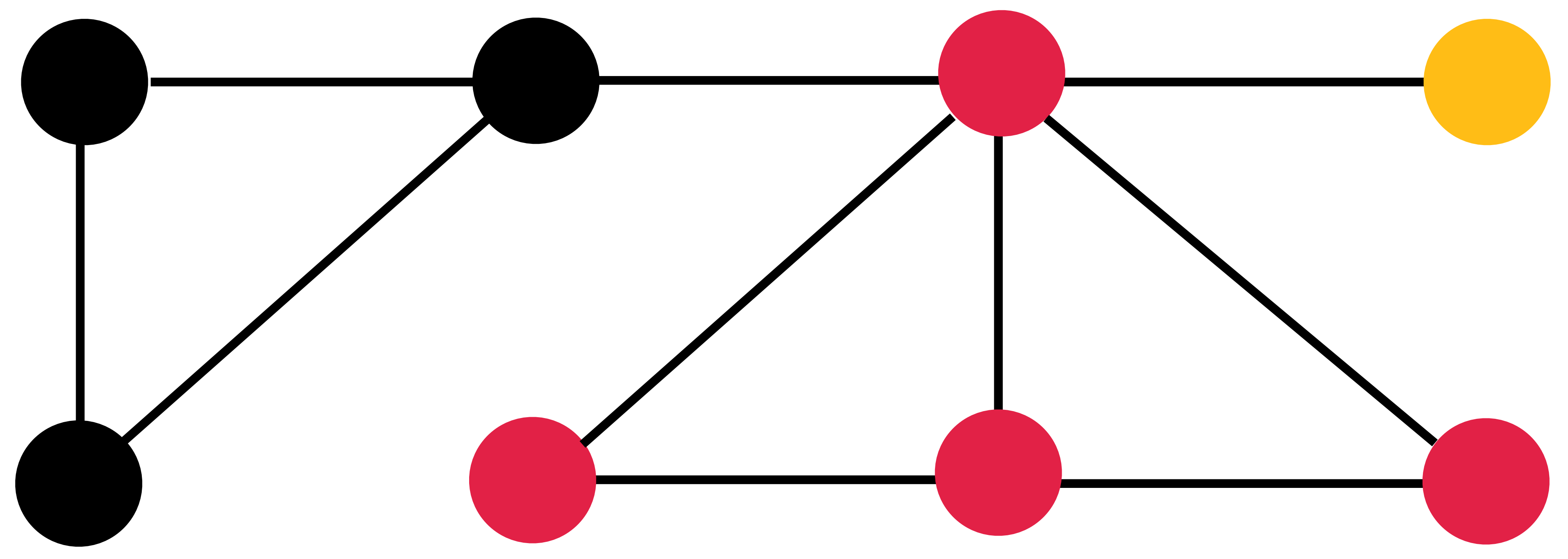
"In simple terms, C is the mean probability that two vertices that are network neighbors of the same other vertex will themselves be neighbors." (Newman, 2013)

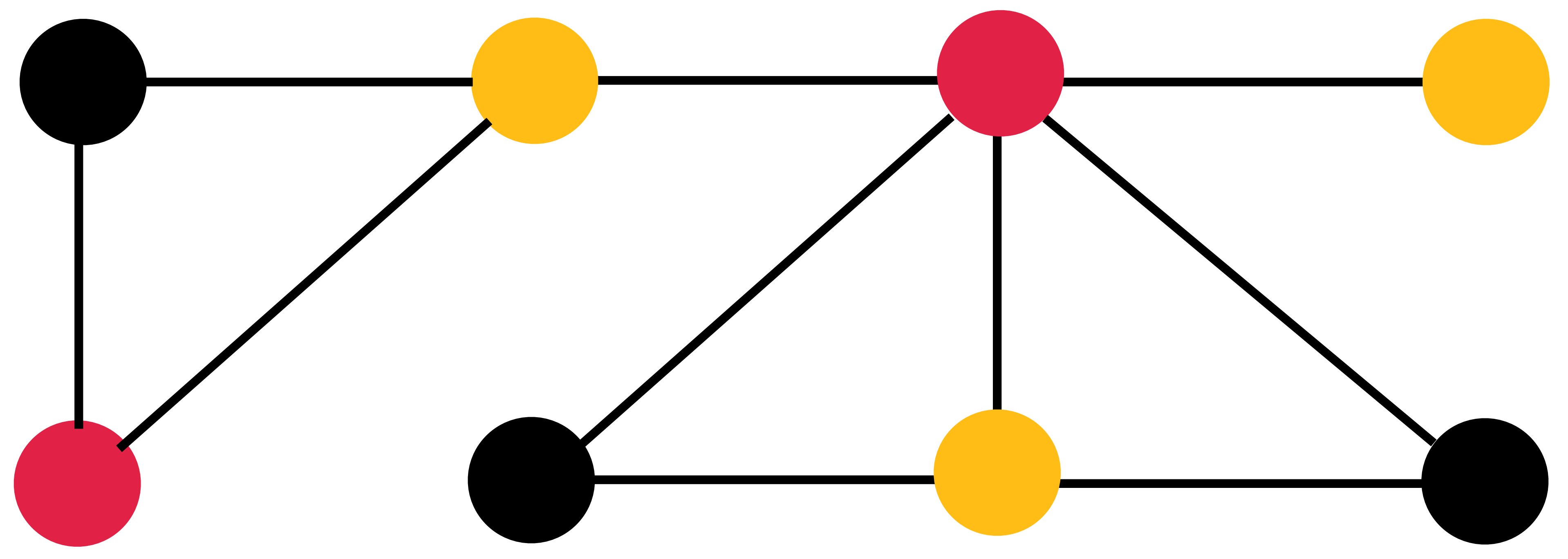
The graph shown above has 1 triangle and 8 connected triples, and so has a clustering coefficient of 3/8 .

There are other ways of defining cluster coefficient but they rely on being able to detect triangles.

Homophily and Heterophily

Homophily and Heterophily





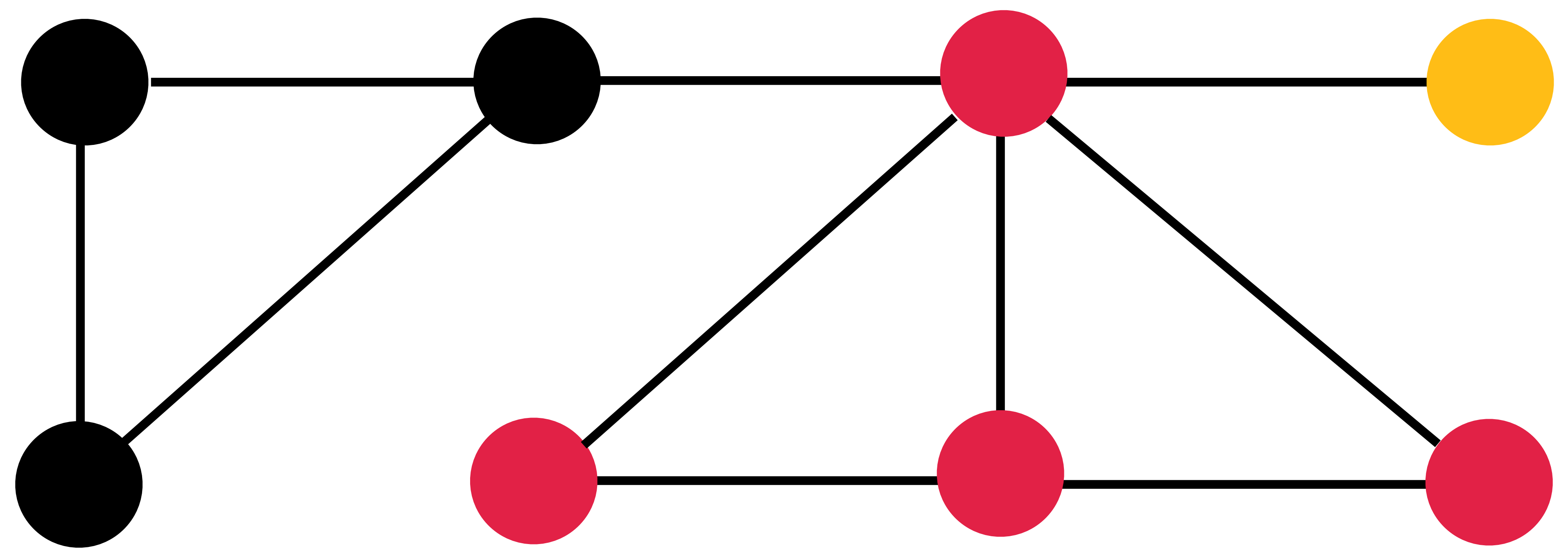
Homophily: Describes a strong positive correlation between nodes and their neighbors within a graph, i.e., a node is highly likely to share features and attributes with its neighbors in the graph.

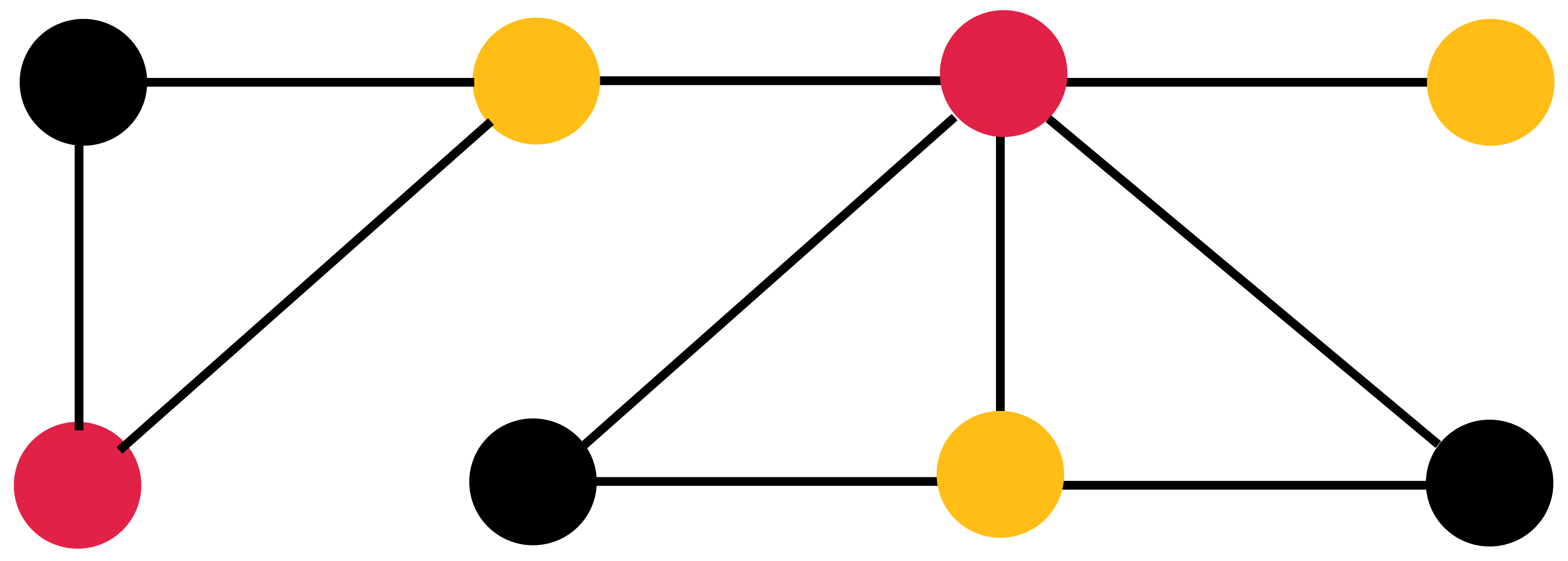
Example: Citation networks, where connected papers tend to tackle similar research areas.

Heterophily: Describes negative correlations between nodes and their neighbors, i.e., a node tends to have contrasting features relative to its neighbors.

Example: Protein graphs, as the proteins interacting with each other may differ from a composition perspective.

Homophily and Heterophily





Data-driven inductive bias: Unlike permutation-invariance, the bias does not rely on structural properties of graphs, but on the application domain and the specific input instances.

Practical: These biases are prominent in real-world applications, and are commonly exploited.

Local vs global: MPNNs employ local operations and neighbor aggregation. Easy to capture correlations by simply adjusting combination and aggregation weights.

Higher-order models are more global: k-GNN requires non-uniform handling of its tuples, based on local neighborhoods, and k-IGN processes all nodes simultaneously, and so must learn to filter out non-local features.

Summary

- The WL hierarchy and its relevance to GNNs
- Higher-order graph neural networks
    - Higher-order message passing neural networks: k-GNNs, hierarchical variants, limitations
    - Invariant/Equivariant graph networks: universality, limitations
    - Provably powerful graph neural networks: expressive power, scalability
- Lack of expressive power may not surface in existing benchmarks.
- Homophily and heterophily: MPNNs vs higher-order models
- There are other extensions of MPNNs, particularly with random features, yielding more expressive power without the need for higher-order tensors — Lecture 7.

References

- H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman. Provably powerful graph networks. *NeurIPS*, 2019a.
- H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman. Invariant and equivariant graph networks. *ICLR*, 2019b.
- H. Maron, E. Fetaya, N. Segol, and Y. Lipman. On the universality of invariant networks. *ICML*, 2019c.
- N. Keriven and G. Peyré, (2019). Universal invariant and equivariant graph neural networks. *NeurIPS*, 2019.
- C. Morris, M. Ritzert, M. Fey, W. Hamilton,J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. *AAAI*, 2019.
- G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.
- Martin Grohe. Descriptive Complexity, Canonisation, and Definable Graph Structure Theory. *Cambridge University Press,* 2017.
- Mark E. J. Newman. The structure and function of complex networks. *SIAM Review*, 2003.
- Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, Thomas Lukasiewicz, The Surprising Power of Graph Neural Networks with Random Node Initialization, IJCAI, 2021
- M. Grohe, The Logic of Graph Neural Networks, *LICS*, 2021.