Chapter 5
The Expressive Power of Graph Neural Networks

Pan Li and Jure Leskovec

Abstract The success of neural networks is based on their strong expressive power that allows them to approximate complex non-linear mappings from features to predictions. Since the universal approximation theorem by (Cybenko, 1989), many studies have proved that feed-forward neural networks can approximate any function of interest. However, these results have not been applied to graph neural networks (GNNs) due to the inductive bias imposed by additional constraints on the GNN parameter space. New theoretical studies are needed to better understand these constraints and characterize the expressive power of GNNs.
In this chapter, we will review the recent progress on the expressive power of GNNs in graph representation learning. We will start by introducing the most widely-used GNN framework— message passing— and analyze its power and limitations. We will next introduce some recently proposed techniques to overcome these limitations, such as injecting random attributes, injecting deterministic distance attributes, and building higher-order GNNs. We will present the key insights of these techniques and highlight their advantages and disadvantages.

5.1 Introduction

Machine learning problems can be abstracted as learning a mapping f^* from some feature space to some target space. The solution to this problem is typically given by a model f_θ that intends to approximate f^* via optimizing some parameter θ. In practice, the ground truth f^* is a priori typically unknown. Therefore, one may expect the model f_θ to approximate a rather broad range of f^*. An estimate of

Pan Li
Department of Computer Science, Purdue University, e-mail: panli@purdue.edu

Jure Leskovec
Department of Computer Science, Stanford University, e-mail: jure@cs.stanford.edu

how broad such a range could be, called the model's *expressive power*, provides an important measure of the model potential. It is desirable to have models with a more expressive power that may learn more complex mapping functions.

Neural networks (NNs) are well known for their great expressive power. Specifically, Cybenko (1989) first proved that any continuous function defined over a compact space could be uniformly approximated by neural networks with sigmoid activation functions and only one hidden layer. Later, this result got generalized to any squashing activation functions by (Hornik et al, 1989).

However, these seminal findings are insufficient to explain the current unprecedented success of NNs in practice because their strong expressive power only demonstrates that the model f_θ is able to approximate f^* but does not guarantee that the model obtained via training $\hat{f}$ indeed approximates f^*. Fig. 5.1 illustrates a well-known curve of Amount of Data vs. Performance of machine learning models (Ng, 2011). NN-based methods may only outperform traditional methods given sufficient data. One important reason is that NNs as machine learning models are still governed by the fundamental tradeoff between the data amount and model complexity (Fig. 5.2). Although NNs could be rather expressive, they are likely to overfit the training examples when paired with more parameters. Therefore, it is necessary for practice to build NNs that can maintain strong expressive power while constraints are imposed on their parameters. At the same time, a good theoretical understanding of the expressive power of NNs with constraints on their parameters is needed.

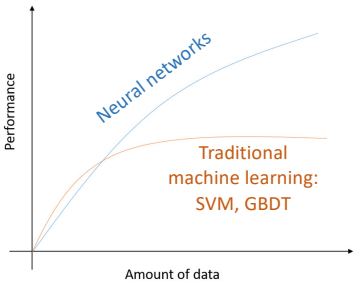


Fig. 5.1: Amount of Data *vs.* Performance of different models.

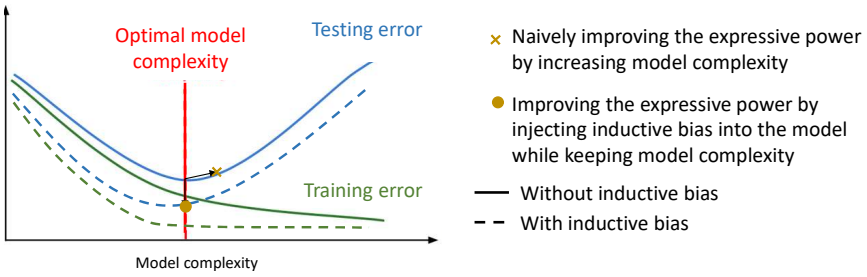


Fig. 5.2: Training and testing errors with and without inductive bias can dramatically affect the expressive power of models.

In practice, constraints on parameters are typically obtained from our prior knowledge of the data; these are referred to as inductive biases. Some significant

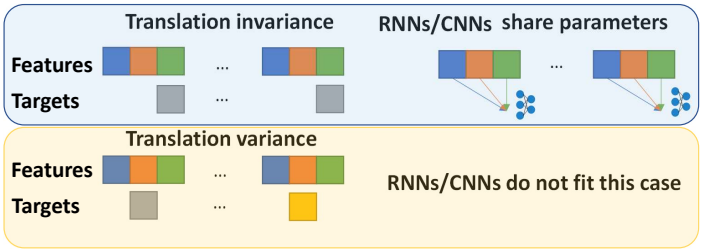


Fig. 5.3: Illustration of 1-dimensional translation invariance/variance. RNNs/CNNs use translation invariance to share parameters.

results about the expressive power of NNs with inductive bias have been shown recently. Yarotsky (2017); Liang and Srikant (2017) have proved that deep neural networks (DNNs), by stacking multiple hidden layers, can achieve good enough approximation with significantly fewer parameters than shallow NNs. The architecture of DNNs leverages the fact that data has typically a hierarchical structure. DNNs are agnostic to the type of data, while dedicated neural network architectures have been developed to support specific types of data. Recurrent neural networks (RNNs) (Hochreiter and Schmidhuber, 1997) or convolution neural networks (CNNs) (LeCun et al, 1989) were proposed to process time series and images, respectively. In these two types of data, effective patterns typically hold translation invariance in time and in space, respectively. To match this invariance, RNNs and CNNs adopt the inductive bias that their parameters have shared across time and space (Fig. 5.3). The parameter-sharing mechanism works as a constraint on the parameters and limits the expressive power of RNNs and CNNs. However, RNNs and CNNs have been shown to have sufficient expressive power to learn translation invariant functions (Siegelmann and Sontag, 1995; Cohen and Shashua, 2016; Khrulkov et al, 2018), which leads to the great practical success of RNNs and CNNs in processing time series and images.

Recently, many studies have focused on a new type of NNs, termed graph neural networks (GNNs) (Scarselli et al, 2008; Bruna et al, 2014; Kipf and Welling, 2017a; Bronstein et al, 2017; Gilmer et al, 2017; Hamilton et al, 2017b; Battaglia et al, 2018). These aim to capture the inductive bias of graphs/networks, another important type of data. Graphs are commonly used to model complex relations and interactions between multiple elements and have been widely used in machine learning applications, such as community detection, recommendation systems, molecule property prediction, and medicine design (Fortunato, 2010; Fouss et al, 2007; Pires et al, 2015). Compared to time series and images, which are well-structured and represented by tables or grids, graphs are irregular and thus introduce new challenges. A fundamental assumption behind machine learning on graphs is that the targets for prediction should be invariant to the order of nodes of the graph. To match this assumption, GNNs hold a general inductive bias termed permutation invariance. In particular, the output given by GNNs should be independent of how the node indices of a graph are assigned and thus in which order are they processed. GNNs require

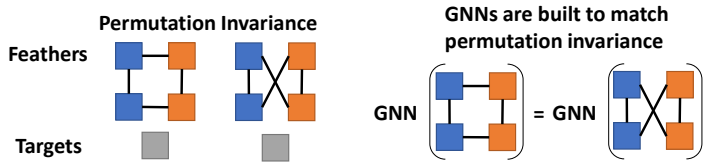


Fig. 5.4: This illustrates how GNNs are designed to maintain permutation invariance.

their parameters to be independent from the node ordering and are shared across the entire graph (Fig. 5.4). Because of this new parameter sharing mechanism in GNNs, new theoretical tools are needed to characterize their expressive power.

Analyzing the expressive power of GNNs is challenging, as this problem is closely related to some long-standing problems in graph theory. To understand this connection, consider the following example of how a GNN would predict whether a graph structure corresponds to a valid molecule. The GNN should be able to identify whether this graph structure is the same, similar, or very different from the graph structures that are known to correspond to valid molecules. Measuring whether two graphs have the same structure involves addressing the graph isomorphism problem, in which no P solutions have yet been found (Helfgott et al, 2017). In addition, measuring whether two graphs have a similar structure requires contending with the graph edit distance problem, which is even harder to address than the graph isomorphism problem (Lewis et al, 1983).

Great progress has been made recently on characterizing the expressive power of GNNs, especially on how to match their power with traditional graph algorithms and how to build more powerful GNNs that overcome the limitation of those algorithms. We will delve more into these recent efforts further along in this chapter. In particular, compared to previous introductions (Hamilton, 2020; Sato, 2020), we will focus on recent key insights and techniques that yield more powerful GNNs. Specifically, we will introduce standard message-passing GNNs that are able to achieve the limit of the 1-dimensional Weisfeiler-Lehman test (Weisfeiler and Leman, 1968), a widely-used algorithm to test for graph isomorphism. We will also discuss a number of strategies to overcome the limitations of the Weisfeiler-Lehman test — including attaching random attributes, attaching deterministic distance attributes, and leveraging higher-order structures.

In Section 5.2, we will formulate the graph representation learning problems that GNNs target. In Section 5.3, we will review the most widely used GNN framework, the message passing neural network, describing the limitations of its expressive power and discussing its efficient implementations. In Section 5.4, we will introduce a number of methods that make GNNs more powerful than the message passing neural network. In Section 5.5, we will conclude this chapter by discussing further research directions.

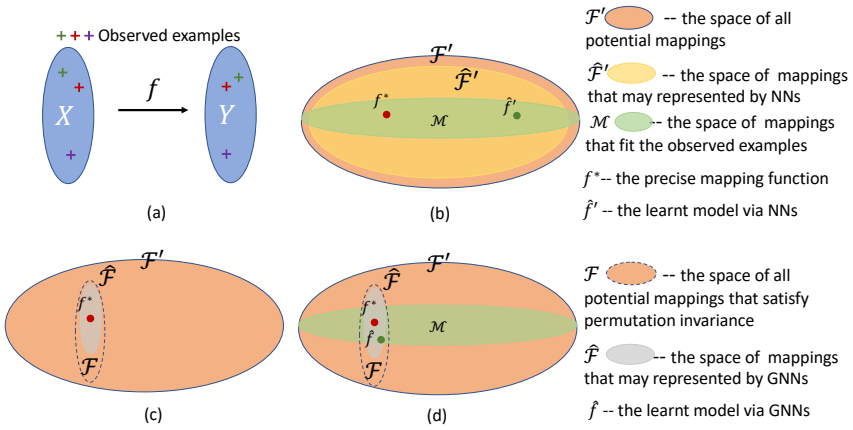


Fig. 5.5: An illustration of the expressive power of NNs and GNNs and their affects on the performance of learned models. a) Machine learning problems aim to learn the mapping from the feature space to the target space based on several observed examples. b) The expressive power of NNs refers to the gap between the two spaces $\mathscr{F}$ and $\hat{\mathscr{F}}'$. Although NNs are expressive ($\hat{\mathscr{F}}'$ is dense in $\mathscr{F}$), the learned model f' based on NNs may differ significantly from f^* due to their overfit of the limited observed data. c) Suppose f^* is known to be permutation invariant, as it works on graph-structured data. Then, the space of potential mapping functions is reduced from $\mathscr{F}'$ to a much smaller space $\mathscr{F}$ that only includes permutation invariant functions. If we adopt GNNs, the space of mapping functions that can be approximated simultaneously reduces to $\hat{\mathscr{F}}$. The gap between $\mathscr{F}$ and $\hat{\mathscr{F}}$ characterizes the expressive power of GNNs. d) Although GNNs are less expressive than general NNs ($\hat{\mathscr{F}} \subset \hat{\mathscr{F}}'$), the learned model based on GNNs f is a much better approximator of f^* as opposed to the one based on NNs $\hat{f}'$. Therefore, for graph-structured data, our understanding of the expressive power of GNNs, *i.e.,* the gap between $\mathscr{F}$ and $\hat{\mathscr{F}}$, is much more relevant than that of NNs.

5.2 Graph Representation Learning and Problem Formulation

In this section, we will set up the formal definition of graph representation learning problems, their fundamental assumption, and their inductive bias. We will also discuss relationships between different notions of graph representation learning problems frequently studied in recent literature.

First, we will start by defining graph-structured data.

Definition 5.1. (Graph-structured data) Let $\mathscr{G} = (\mathscr{V}, \mathscr{E}, X)$ denote an attributed graph, where $\mathscr{V}$ is the node set, $\mathscr{E}$ is the edge set, and $X \in \mathbb{R}^{|\mathscr{V}| \times F}$ are the node attributes. Each row of X, $X_v \in \mathbb{R}^F$ refers to the attributes on the node $v \in \mathscr{V}$. In practice, graphs are usually sparse, *i.e.*, $|\mathscr{E}| \ll |\mathscr{V}|^2$. We introduce $A \in \{0,1\}^{|\mathscr{V}| \times |\mathscr{V}|}$ to denote the adjacency matrix of G such that $A_{uv} = 1$ iff $(u,v) \in E$. Combining the

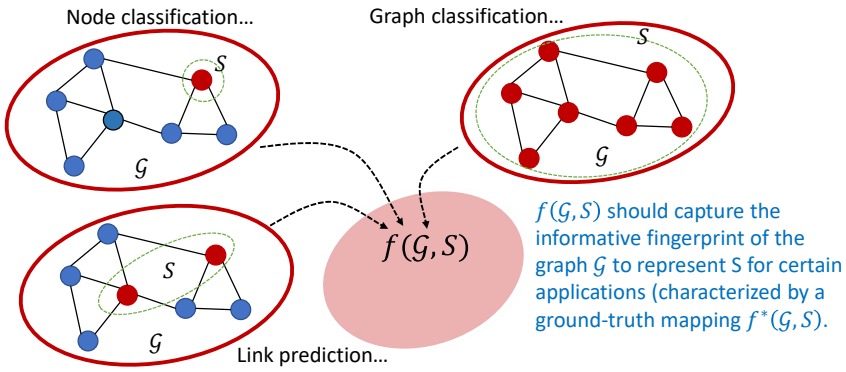


Fig. 5.6: Graph representation learning problems frequently discussed in literature.

adjacency matrix and node attributes, we may also denote $\mathscr{G} = (A, X)$. Moreover, if $\mathscr{G}$ is unattributed with no node attributes, we can assume that all elements in X are constant. Later, we also use $\mathscr{V}[\mathscr{G}]$ to denote the entire node set of a particular graph $\mathscr{G}$.

The goal of graph representation learning is to learn a model by taking graph-structured data as input and then mapping it so that certain prediction targets are matched. Different graph representation learning problems may apply to a varying number of nodes in a graph. For example, for node classification, a prediction is made for each node, for each link/relation prediction on a pair of nodes, and for each graph classification or graph property prediction on the entire node set V. We can unify all these problems as graph representation learning.

Definition 5.2. (Graph representation learning) The feature space is defined as $\mathscr{X} := \Gamma \times \mathscr{S}$, where Γ is the space of graph-structured data and $\mathscr{S}$ includes all the node subsets of interest, given a graph $\mathscr{G} \in \Gamma$. Then, a point in $\mathscr{X}$ can be denoted as $(\mathscr{G}, S)$, where S is a subset of nodes that are in $\mathscr{G}$. Later, we call $(\mathscr{G}, S)$ as a graph representation learning (GRL) example. Each GRL example $(\mathscr{G}, S) \in \mathscr{X}$ is associated with a target y in the target space $\mathscr{Y}$. Suppose the ground-truth association function between features and targets is denoted by $f^* : \mathscr{X} \to \mathscr{Y}$, $f^*(\mathscr{G}, S) = y$. Given a set of training examples $\Xi = \{(\mathscr{G}^{(i)}, S^{(i)}, y^{(i)})\}_{i=1}^k$ and a set of testing examples $\Psi = \{(\tilde{\mathscr{G}}^{(i)}, \tilde{S}^{(i)}, \tilde{y}^{(i)})\}_{i=1}^k$, a *graph representation learning* problem is to learn a function f based on Ξ such that f is close to f^* on Ψ.

The above definition is general in the sense that in a GRL example $(\mathscr{G}, S) \in \mathscr{X}$, $\mathscr{G}$ provides both raw and structural features on which some prediction for a node subset S of interest is to be made. Below, we will further list a few frequently-investigated learning problems that may be formulated as graph representation learning problems.

Remark 5.1. (Graph classification problem / Graph-level prediction) The node set S of interest is the entire node set $\mathscr{V}[\mathscr{G}]$ by default. The space of graph-structured data

Γ typically contains multiple graphs. The target space $\mathscr{Y}$ contains labels of different graphs. Later, for graph-level prediction, we will use $\mathscr{G}$ to denote a GRL example instead of $(\mathscr{G},S)$ for notational simplicity.

Remark 5.2. (Node classification problem / Node-level prediction) In a GRL example $(\mathscr{G},S)$, the S corresponds to one single node of interest. The corresponding $\mathscr{G}$ can be defined in different ways. On the one hand, only the nodes close to S provide effective features. In this case, $\mathscr{G}$ may be set as the induced local subgraph around S. Different $\mathscr{G}$'s for different S's may come from a single graph. On the other hand, two nodes that are far apart on one graph still hold mutual impact and can be used as a feature to make a prediction on another graph. In that case, $\mathscr{G}$ needs to include a large portion of a graph or even the entire graph.

Remark 5.3. (Link prediction problem / Node-pair-level prediction) In a GRL example $(\mathscr{G},S)$, S corresponds to a pair of nodes of interest. Similar to the node classification problem, $\mathscr{G}$ for each example may be an induced subgraph around S or the entire graph. The target space $\mathscr{Y}$ contains 0-1 labels that indicate whether there is a probable link between two nodes. $\mathscr{Y}$ may also be generalized to include labels that reflect the types of links to be predicted.

Next, we will introduce the fundamental assumption used in most graph representation learning problems.

Definition 5.3. (Isomorphism) Consider two GRL examples $(\mathscr{G}^{(1)},S^{(1)})$, $(\mathscr{G}^{(2)},S^{(2)}) \in \mathscr{X}$. Suppose $\mathscr{G}^{(1)} = (A^{(1)},X^{(1)})$ and $\mathscr{G}^{(2)} = (A^{(2)},X^{(2)})$. If there exists a bijective mapping $\pi : \mathscr{V}[\mathscr{G}^{(1)}] \to \mathscr{V}[\mathscr{G}^{(2)}]$, $i \in \{1,2\}$, such that $A^{(1)}_{uv} = A^{(2)}_{\pi(u)\pi(v)}$, $X^{(1)}_{u} = X^{(2)}_{\pi(u)}$ and π also gives a bijective mapping between $S^{(1)}$ and $S^{(2)}$, we call that $(\mathscr{G}^{(1)},S^{(1)})$ and $(\mathscr{G}^{(2)},S^{(2)})$ are isomorphic, denoted as $(\mathscr{G}^{(1)},S^{(1)}) \cong (\mathscr{G}^{(2)},S^{(2)})$. When the particular bijective mapping π should be highlighted, we use notation $(\mathscr{G}^{(1)},S^{(1)}) \overset{\pi}{\cong} (\mathscr{G}^{(2)},S^{(2)})$. If there is no such a π, we call that they are non-isomorphic, denoted as $(\mathscr{G}^{(1)},S^{(1)}) \ncong (\mathscr{G}^{(2)},S^{(2)})$.

Assumption 1 *(Fundamental assumption in graph representation learning) Consider a graph representation learning problem with a feature space $\mathscr{X}$ and its corresponding target space $\mathscr{Y}$. Pick any two GRL examples $(\mathscr{G}^{(1)},S^{(1)})$, $(\mathscr{G}^{(2)},S^{(2)}) \in \mathscr{X}$. The fundamental assumption says that if $(\mathscr{G}^{(1)},S^{(1)}) \cong (\mathscr{G}^{(2)},S^{(2)})$, their corresponding targets in $\mathscr{Y}$ are the same.*

Due to this fundamental assumption, it is natural to introduce the corresponding *permutation invariance* as inductive bias that all models of graph representation learning should satisfy.

Definition 5.4. (Permutation invariance) A model f satisfies permutation invariance if for any $(\mathscr{G}^{(1)},S^{(1)}) \cong (\mathscr{G}^{(2)},S^{(2)})$, $f(\mathscr{G}^{(1)},S^{(1)}) = f(\mathscr{G}^{(2)},S^{(2)})$.

Now we may define the expressive power of a model for graph representation learning problems.

Definition 5.5. (Expressive power) Consider a feature space $\mathscr{X}$ of a graph representation learning problem and a model f defined on $\mathscr{X}$. Define another space $\mathscr{X}(f)$ as a subspace of the quotient space $\mathscr{X}/\cong$ such that for two GRL examples $(\mathscr{G}^{(1)},S^{(1)})$, $(\mathscr{G}^{(2)},S^{(2)}) \in \mathscr{X}(f)$, $f(\mathscr{G}^{(1)},S^{(1)}) \neq f(\mathscr{G}^{(2)},S^{(2)})$. Then, the size of $\mathscr{X}(f)$ characterizes the expressive power of f. For two models, $f^{(1)}$ and $f^{(2)}$, if $\mathscr{X}(f^{(1)}) \supset \mathscr{X}(f^{(2)})$, we say that $f^{(1)}$ is more expressive than $f^{(2)}$.

Remark 5.4. Note that the expressive power in Def. 5.5, characterized by how a model can distinguish non-isomorphic GRL examples, does not exactly match the traditional expressive power used for NNs in the sense of functional approximation. Actually, Def. 5.5 is strictly weaker because distinguishing any non-isomorphic GRL examples does not necessarily indicate that we can approximate any function f^* defined over $\mathscr{X}$. However, if a model f cannot distinguish two non-isomorphic features, f is definitely unable to approximate function f^* that maps these two examples to two different targets. Some recent studies have been able to prove some equivalence between distinguishing non-isomorphic features and permutation invariant function approximations under weak assumptions and applying involved techniques (Chen et al, 2019f; Azizian and Lelarge, 2020). Interested readers may check these references for more details.

It is trivial to provide the expressive power of a model f for graph representation learning if f does not satisfy permutation invariance. Without such a constraint, NNs can approximate all continuous functions (Cybenko, 1989), which include the continuous functions that distinguish any non-isomorphic GRL examples. Therefore, the key question we are to discuss in the chapter is: *"How to build the most expressive permutation invariant models, GNNs in particular, for graph representation learning problems?"*

5.3 The Power of Message Passing Graph Neural Networks

5.3.1 Preliminaries: Neural Networks for Sets

We will start by reviewing the NNs with sets (multisets) as their input,since a set can be viewed as a simplified-version of a graph where all edges are removed. By definition, the order of elements of a set does not impact the output; models that encode sets naturally provide an important building block for encoding the graphs. We term this approach invariant pooling.

Definition 5.6. (Multiset) A multiset is a set where its elements can be repetitive, meaning that they are present multiple times. In this chapter, we assume by default that all the sets are multisets and thus allow repetitive elements. In situations where this is not the case, we will indicate otherwise.

Definition 5.7. (Invariant pooling) Given a multiset of vectors $S = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_k\}$ where $\mathbf{a}_i \in \mathbb{R}^F$ and F is an arbitrary constant, an invariant pooling refers to a mapping, denoted as $q(S)$, that is invariant to the order of elements in S.

Some widely-used invariant pooling operations include: sum pooling $q(S) = \sum_{i=1}^{k} \mathbf{a}_i$, mean pooling $q(S) = \frac{1}{k}\sum_{i=1}^{k} \mathbf{a}_i$ and max pooling $[q(S)]_j = \max_{i\in[1,F]}\{a_{ij}\}$ for all $j \in [1,F]$. Zaheer et al (2017) show that any invariant poolings of a set S can be approximated by $q(S) = \phi(\sum_{i=1}^{k} \psi(\mathbf{a}_i))$, where ϕ and ψ are functions that may be approximated by fully connected NNs, provided that $\mathbf{a}_i$, $i \in [k]$ comes from a countable universe. This statement can be generalized to the case where S is a multiset (Xu et al, 2019d).

5.3.2 Message Passing Graph Neural Networks

Message passing is the most widely-used framework to build GNNs (Gilmer et al, 2017). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$, the message passing framework encodes each node $v \in \mathcal{V}$ with a vector representation $\mathbf{h}_v$ and keeps updating this node representation by iteratively collecting representations of its neighbors and applying neural network layers to perform a non-linear transformation of those collections:

1. Initialize node vector representations as node attributes: $\mathbf{h}_v^{(0)} \leftarrow X_v, \forall v \in \mathcal{V}$.
2. Update each node representation based on message passing over the graph structure. In l-th layer, $l = 1, 2, ..., L$, perform the following steps:

$$\text{Message:} \quad \mathbf{m}_{vu}^{(l)} \leftarrow \text{MSG}(\mathbf{h}_v^{(l-1)}, \mathbf{h}_u^{(l-1)}), \ \forall (u,v) \in \mathcal{E}, \tag{5.1}$$

$$\text{Aggregation:} \quad \mathbf{a}_v^{(l)} \leftarrow \text{AGG}(\{\mathbf{m}_{vu}^{(l)} | u \in \mathcal{N}_v\}), \ \forall v \in \mathcal{V}, \tag{5.2}$$

$$\text{Update:} \quad \mathbf{h}_v^{(l)} \leftarrow \text{UPT}(\mathbf{h}_v^{(l-1)}, \mathbf{a}_v^{(l)}), \ \forall v \in \mathcal{V}. \tag{5.3}$$

where $\mathcal{N}_v$ is the set of neighbors of v.

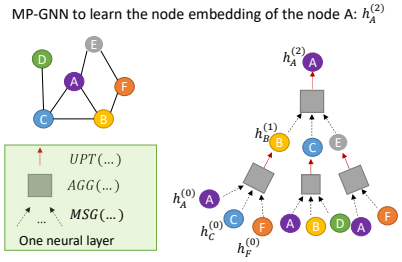


Fig. 5.7: The computing flow of MP-GNN to obtain a node representation.

The operations MSG, AGG, and UPT can be implemented via neural networks. Typically, MSG is implemented by a feed-forward NN, *e.g.,* $\text{MSG}(p,q) = \sigma(pW_1 + qW_2)$, where W_1 and W_2 are learnable weights, and $\sigma(\cdot)$ is an element-wise non-linear activation. UPT can be chosen in a similar way as MSG. AGG differs as its input is a multiset of vectors and thus the order of these vectors should not affect the output. AGG is typically implemented as an invariant pooling (Def. 5.7). Each layer k can have different parameters from other layers. We will denote the GNNs that follow this message passing framework as *MP-GNN*.

MP-GNN produces representations of all the nodes, $\{\mathbf{h}_v^{(L)}|v \in V\}$. Each node representation is essentially determined by a subtree rooted at this node (Fig. 5.7). Given a specific graph representation learning problem, for example, classifying a set of nodes $S \subseteq V$, we may use the representations of relevant nodes in S to make the prediction:

$$\hat{y}_S = \text{READOUT}(\{\mathbf{h}_v^{(L)}|v \in S\}). \tag{5.4}$$

where the READOUT operation is often implemented via another invariant pooling when $|S| > 1$ plus a feed-forward NN to predict the target. Combining Eqs.equation 11.45-equation 5.4, MP-GNN builds a GNN model for graph representation learning:

$$\hat{y}_S = f_{MP-GNN}(\mathscr{G}, S). \tag{5.5}$$

We can show the permutation invariance of MP-GNN by induction over the iteration index l.

Theorem 5.1. *(Invariance of MP-GNN) $f_{MP-GNN}(\cdot,\cdot)$ satisfies permutation invariance (Def. 5.4) as long as the AGG and READOUT operations are invariant pooling operations (Def. 5.7).*

Proof. This can be proved trivially by induction.

MP-GNN by default leverages the inductive bias that the nodes in the graph directly affect each other only via their connected edges. The mutual effect between nodes that are not connected by an edge can be captured via paths that connect these nodes via message passing. Indeed, such inductive bias may not match the assumptions in a specific application, and MP-GNN may find it hard to capture mutual effect between two far-away nodes. However, the message-passing framework has several benefits for model implementation and practical deployment. First, it directly works on the original graph structure and no pre-processing is needed. Second, graphs in practice are typically sparse ($|\mathscr{E}| \ll |\mathscr{V}|^2$) and thus MP-GNN is able to scale to very large but sparse graphs. Third, each of the three operations MSG, AGG, and UPT can be computed in parallel across all nodes and edges, which is beneficial for parallel computing platforms such as GPUs and map-reduce systems.

Because it is natural and easy to be implemented in practice, most GNN architectures essentially follow the MP-GNN framework by adopting specific MSG, AGG, and UPT operations. Representative approaches include InteractionNet (Battaglia et al, 2016), structure2vec (Dai et al, 2016), GCN (Kipf and Welling, 2017a), GraphSAGE (Hamilton et al, 2017b), GAT (Veličković et al, 2018), GIN (Xu et al, 2019d), and many others (Kearnes et al, 2016; Zhang et al, 2018g).

5.3.3 The Expressive Power of MP-GNN

In this section, we will introduce the expressive power of MP-GNN , following the results proposed in Xu et al (2019d); Morris et al (2019).

The 1-dimensional Weisfeiler-Lehman test to distinguish $(\mathcal{G}^{(1)}, S^{(1)})$ and $(\mathcal{G}^{(2)}, S^{(2)})$:

1. Assume each node v in $\mathcal{V}[\mathcal{G}^{(i)}]$ is initialized with a color $C_v^{(i,0)} \leftarrow X_v^{(i)}$ for $i = 1,2$. If $X_v^{(i)}$ is a vector, then an injective function is used to map it to a color.
2. For $l = 1,2,...,$ do

$$\textbf{Update node colors:} \quad C_v^{(i,l)} \leftarrow \text{HASH}(C_v^{(i,l-1)}, \{C_u^{(i,l-1)} | u \in \mathcal{N}_v^{(i)}\}), \quad i \in \{1,2\} \tag{5.6}$$

where the HASH operation can be viewed as an injective mapping where different tuples $(C_v^{(i,l-1)}, \{C_u^{(i,l-1)} | u \in \mathcal{N}_v^{(i)}\})$ are mapped to different labels.

Test: If two multisets $\{C_v^{(1,l)} | v \in S^{(1)}\}$ and $\{C_v^{(2,l)} | v \in S^{(2)}\}$ are not equal, then return $(\mathcal{G}^{(1)}, S^{(1)}) \ncong (\mathcal{G}^{(2)}, S^{(2)})$; else, go back to equation 5.6.

If 1-WL test returns $(\mathcal{G}^{(1)}, S^{(1)}) \ncong (\mathcal{G}^{(2)}, S^{(2)})$, we know that $(\mathcal{G}^{(1)}, S^{(1)})$ $(\mathcal{G}^{(2)}, S^{(2)})$ are not isomorphic. However, for some non-isomorphic $(\mathcal{G}^{(1)}, S^{(1)})$ $(\mathcal{G}^{(2)}, S^{(2)})$, the 1-WL test may not return $(\mathcal{G}^{(1)}, S^{(1)}) \ncong (\mathcal{G}^{(2)}, S^{(2)})$ (even with infinitely many iterations). In this case, the 1-WL test fails to distinguish them. Note that the 1-WL test was originally proposed to test the isomorphism of two entire graphs, *i.e.,*, $S^{(i)} = \mathcal{V}[\mathcal{G}^{(i)}]$ for $i \in \{1,2\}$ (Weisfeiler and Leman, 1968). Here the 1-WL test is further generalized to test the case where $S^{(i)} \subset \mathcal{V}^{(i)}$, to match the general graph representation learning problems.

The expressive power we defined (Def. 5.5) is closely related to the graph isomorphism problem. This problem is challenging, as no polynomial-time algorithms have been found for it (Garey, 1979; Garey and Johnson, 2002; Babai, 2016). Despite some corner cases (Cai et al, 1992), the Weisfeiler-Lehman (WL) tests of graph isomorphism (Weisfeiler and Leman, 1968) are a family of effective and computationally efficient tests that distinguish a broad class of graphs (Babai and Kucera, 1979). Its 1-dimensional form (the 1-WL test), "naive vertex refinement", is analogous to the neighborhood aggregation in MP-GNN .

They are comparing MP-GNN with the 1-WL test, the node-representation updating procedure Eqs.equation 11.45-equation 5.3 can be viewed as an implementation of Eq.equation 5.6 and the READOUT operation in Eq.equation 5.4 can be viewed as a summary of all node representations. Although MP-GNN was not proposed to perform graph isomorphism testing, the f_{MP-GNN} can be used for this test: if $f_{MP-GNN}(\mathcal{G}^{(1)}, S^{(1)}) \neq f_{MP-GNN}(\mathcal{G}^{(2)}, S^{(2)})$, then we know that $(\mathcal{G}^{(1)}, S^{(1)}) \ncong (\mathcal{G}^{(2)}, S^{(2)})$. Because of this analogy, the expressive power of MP-GNN can be measured by the 1-WL test. Formally, we conclude the argument in the following theorem.

Theorem 5.2. *(Lemma 2 in (Xu et al, 2019d), Theorem 1 in (Morris et al, 2019)) Consider two non-isomorphic GRL examples $(\mathcal{G}^{(1)}, S^{(1)})$ and $(\mathcal{G}^{(2)}, S^{(2)})$. If $f_{MP-GNN}(\mathcal{G}^{(1)}, S^{(1)}) \neq f_{MP-GNN}(\mathcal{G}^{(2)}, S^{(2)})$, then the 1-WL test also decides $(\mathcal{G}^{(1)}, S^{(1)})$ and $(\mathcal{G}^{(2)}, S^{(2)})$ are not isomorphic.*

Theorem 5.2 indicates that MP-GNN is at most as powerful as the 1-WL test in distinguishing different graph-structured features. Here, the 1-WL test is considered an upper bound (instead of being equal to the expressive power of MP-GNN)

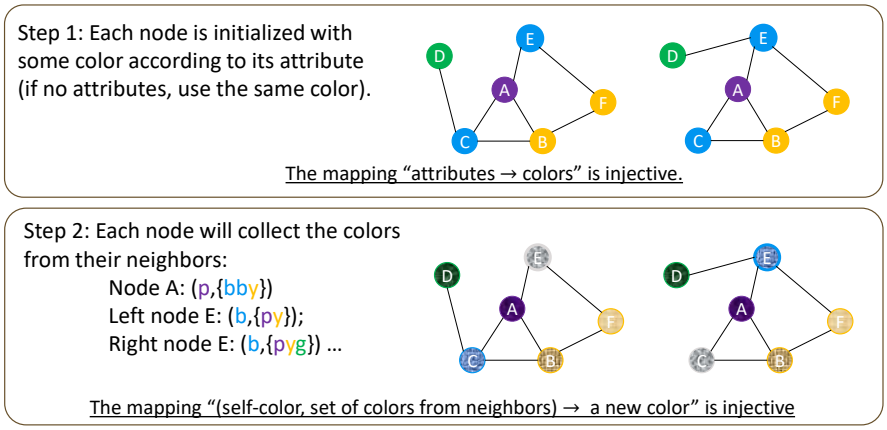


Fig. 5.8: An illustration of steps that distinguish two graphs via the 1-dimensional Weisfeiler-Lehman test. MP-GNN follows a similar procedure and may also distinguish them.

because the updating procedure which aggregates node colors from its neighbors (Eq.equation 5.6) is injective and can distinguish between the different aggregations of node colors. This intuition is useful later to design MP-GNN that matches this upper bound.

Now that the upper bound of the representation power of MP-GNN has been established, a natural follow-up question is whether there are existing GNNs that are, in principle, as powerful as the 1-WL test. The answer is yes. As shown by Theorem 5.3, if the message passing operation (compositing Eqs.equation 11.45-equation 5.3 together) and the final READOUT (Eq.equation 5.4) are both injective, then the resulting MP-GNN is as powerful as the 1-WL test.

Theorem 5.3. *(Theorem 3 in (Xu et al, 2019d)) After sufficient iterations, MP-GNN may map any GRL examples* $(\mathcal{G}^{(1)}, S^{(1)})$ *and* $(\mathcal{G}^{(2)}, S^{(2)})$*, that the 1-WL test decides as non-isomorphic, to different representations if the following two conditions hold:*

a) The composition of MSE, AGG and UPT (Eqs.equation 11.45-equation 5.3) constructs an injective mapping from $(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} | u \in \mathcal{N}_v\})$ *to* $\mathbf{h}_v^{(k)}$.
b) The READOUT (Eq.equation 5.4) is injective.

Although MP-GNN does not surpass the representation power of the 1-WL test, MP-GNN has important benefits over the 1-WL test from the perspective of machine learning: node colors and the final decision given by the 1-WL test are discrete (represented as node colors or a "yes/no" decision) and thus cannot capture the similarity between graph structures. In contrast, a MP-GNN satisfying the criteria in

Theorem 5.3 generalizes the 1-WL test by *learning to represent* the graph structures with vectors in a continuous space. This enables MP-GNN to not only discriminate between different structures but also to learn to map similar graph structures to similar representations, thus capturing dependencies between graph structures. Such learned representations are particularly helpful for generalizations where data contains noisy edges and the exact matching graph structures are sparse (Yanardag and Vishwanathan, 2015).

In the next subsection, we will focus on introducing the key design ideas behind MP-GNN that satisfies the conditions in Theorem 5.3.

5.3.4 MP-GNN with the Power of the 1-WL Test

Xu et al (2019d) introduced the key guidelines to satisfy the conditions in Theorem 5.3. First, to model injective multiset functions for the neighbor aggregation, the AGG operation (Eq.equation 15.16) is suggested to adopt the *sum* pooling operation, which is proved to universally represent functions defined over multisets whose elements are from a countable space (Lemma 5.1).

Lemma 5.1. *(Lemma 4 in (Xu et al, 2019d)) Suppose $\mathscr{S}$ is a countable universe of elements. Then there exists a function $q : \mathscr{S} \to \mathbb{R}^n$ such that $q(S) = \sum_{x \in S} \psi(x)$ is unique for each finite multiset $S \subset \mathscr{S}$, where ψ individually encodes each element in $\mathscr{S}$. Moreover, any multiset function g can be decomposed as $g(S) = \phi\left(\sum_{x \in S} \psi(x)\right)$ for some function ϕ.*

Remark 5.5. Note that the sum pooling operator is crucial, as some popular invariant pooling operators, such as the mean pooling operator, are not injective multiset functions. The significance of the sum pooling operation is to record the number of repetitive elements in a multiset. The mean pooling operation adopted by graph convolutional network (Kipf and Welling, 2017a) or the softmax-normalization (attention) pooling adopted by graph attention network (Veličković et al, 2018) may learn the distribution of the elements in a multiset but not the precise counts of the elements.

Thanks to the universal approximation theorem (Hornik et al, 1989), we can use multi-layer perceptrons (MLPs) to model and learn ψ and ϕ in Lemma 5.1 for universally injective AGG operation. In MP-GNN, we do not even need to explicitly model ψ and ϕ as the MSG and UPT operations — (Eqs.equation 11.45 and equation 5.3) respectfully — have already been implemented via MLPs. Therefore, using the sum pooling as the AGG operation is sufficient to achieve the most expressive MP-GNN:

$$\begin{aligned}
\text{Expressive Message:} \quad & \mathbf{m}_{vu}^{(k)} \leftarrow MLP_1^{(k-1)}(\mathbf{h}_v^{(k-1)} \oplus \mathbf{h}_u^{(k-1)}),\ \forall (u,v) \in \mathscr{E}, \\
\text{Expressive Aggregation:} \quad & \mathbf{a}_v^{(k)} \leftarrow \sum_{u \in \mathscr{N}_v} \mathbf{m}_{vu}^{(k)},\ \forall v \in \mathscr{V}, \\
\text{Expressive Update:} \quad & \mathbf{h}_v^{(k)} \leftarrow MLP_2^{(k-1)}(\mathbf{h}_v^{(k-1)} \oplus \mathbf{a}_v^{(k)}),\ \forall v \in \mathscr{V}.
\end{aligned}$$

where $\oplus$ denotes concatenation. Actually, we can even simplify the procedure by using a single MLP. We can also set $\mathbf{m}_{vu}^{(k)} \rightarrow \mathbf{h}_u^{(k-1)}, \forall (u,v) \in E$ without decreasing the expressive power. Combining all the terms together, Xu et al (2019d) obtains the simplest update mechanism of node representations that constructs an injective mapping from $(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k)} | u \in \mathscr{N}_v\})$ to $\mathbf{h}_v^{(k)}$:

$$\mathbf{h}_v^{(k)} \leftarrow MLP^{(k-1)}((1+\varepsilon^{(k)})\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathscr{N}_v} \mathbf{h}_u^{(k-1)}),\ \forall v \in \mathscr{V}, \tag{5.7}$$

where $\varepsilon^{(k)}$ is a learnable weight. This updating method, by using a NN-based language, is termed the graph isomorphism network (GIN) layer (Xu et al, 2019d).

Lemma 5.2 formally states that MP-GNN that adopts Eq.equation 5.7 may match the condition a) in Theorem 5.3.

Lemma 5.2. *Updating node representations by following Eq.equation 5.7 constructs an injective mapping from $(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k)} | u \in \mathscr{N}_v\})$ to $\mathbf{h}_v^{(k)}$, if the node attributes X are from a countable space.*

Proof. Combine the proof for injectiveness of the sum aggregation with the universal approximation property of MLP (Hornik et al, 1989).

A similar idea may be adapted to the READOUT operation (Eq.5.4), which also requires an injective mapping of multisets:

$$\text{Expressive Inference:} \quad \hat{y}_S = MLP(\sum_{v \in S} \mathbf{h}_v^{(L)}). \tag{5.8}$$

Xu et al (2019d) has observed that node representations from earlier iterations may sometimes generalize better and thus also suggests using the READOUT (a counterpart to Eq.5.4) from the Jumping Knowledge Network (JK-Net) (Xu et al, 2018a), though it is not necessary from the perspective of the representation power of MP-GNN .

Overall, combining the update Eq.equation 5.7 and the READOUT Eq.equation 5.8, we may achieve an MP-GNN that is as powerful as the 1-WL test. In the next section, we introduce several techniques that allow MP-GNN to break the limitation of the 1-WL test and achieve even stronger expressive power.

5.4 Graph Neural Networks Architectures that are more Powerful than 1-WL Test

In the previous section, we characterized the representation power of MP-GNN that is bounded by the 1-WL test. In other words, if the 1-WL test cannot distinguish two GRL examples $(\mathcal{G}^{(1)}, S^{(1)})$ and $(\mathcal{G}^{(2)}, S^{(2)})$, then MP-GNN cannot distinguish them either. Although the 1-WL test cannot distinguish only a few corner graph structures, it indeed limits the applicability of GNNs in many real-world applications (You et al, 2019; Chen et al, 2020q; Ying et al, 2020b). In this section, we will introduce several approaches to overcome the above limitation of MP-GNN.

5.4.1 Limitations of MP-GNN

First, we will review several critical limitations of MP-GNN and the 1-WL test to gain the intuition for understanding the techniques that build more powerful GNNs. MP-GNN iteratively updates the representation of each node by aggregating representations of its neighbors. The obtained node representation essentially encodes the subtree rooted at Node v (Fig. 5.7). However, using this rooted subtree to represent a node may lose useful information, such as:

1. The information about the distance between multiple nodes is lost. For example, You et al (2019) noticed that MP-GNN has limited power in capturing the position/location of a given node with respect to another node in the graph. Many nodes may share similar subtrees, and thus, MP-GNN produces the same representation for them although the nodes may be located at different locations in the graph. This location information of nodes is crucial for the tasks that depend on multiple nodes, such as link prediction (Liben-Nowell and Kleinberg, 2007), as two nodes that tend to be connected with a link are typically located close to each other. An illustrative example is shown in Fig. 5.9.
2. The information about cycles is lost. Particularly, when expanding the subtree of a node, MP-GNN essentially losses track of the node identities in the subtrees. An illustrative example is shown in Fig. 5.10. The information about cycles is crucial in applications such as subgraph matching (Ying et al, 2020b) and counting (Liu et al, 2020e) because loops frequently appear in the queried subgraph patterns of a subgraph matching/counting problem. Chen et al (2020q) formally proved that MP-GNN is able to count star structures (a particular form of trees) but cannot count connected subgraphs with three or more nodes that form cycles.

Theoretically, there is a general class of graph representation learning problems that MP-GNN will fail to solve due to its limited representation. To show this, we define a class of graphs, termed attributed regular graphs.

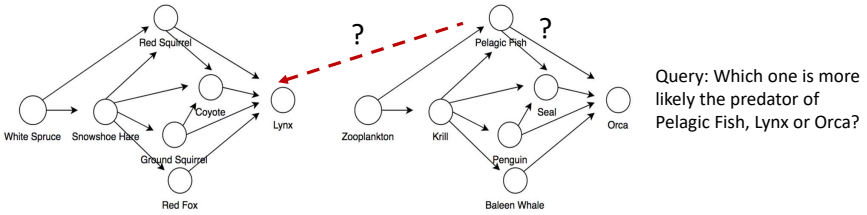


Fig. 5.9: A foodweb network example that demonstrates limitations of MP-GNN (Srinivasan and Ribeiro, 2020a). MP-GNN will associate Lynx and Orca with the same node representations, *i.e.,* $\mathbf{h}_{\text{Lynx}}^{(i)} = \mathbf{h}_{\text{Orca}}^{(i)}$, as these two nodes hold the same rooted subtree. Note that we do not consider node features. Thus, MP-GNN cannot predict whether Lynx or Orca is more likely to be the predator of Pelagic Fish (a link prediction task).

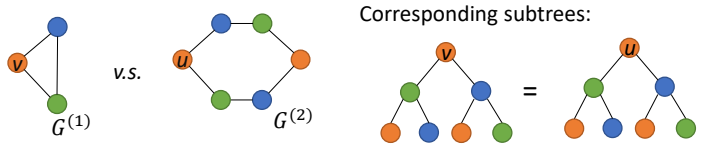


Fig. 5.10: The node representations $\mathbf{h}_v^{(L)}$ and $\mathbf{h}_u^{(L)}$ given by MP-GNN are the same, although they belong to different cycles – 3-cycle and 6-cycle, respectively.

Definition 5.8. (Attributed regular graphs) Consider an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$. All nodes in $\mathcal{V}$ are partitioned according to their attributes $\mathcal{V} = \cup_{i=1}^{k} V_i$, such that two nodes from the same category V_i have the same attributes, while two nodes from different categories have different attributes. If for any two categories, V_i, V_j, $i, j \in [k]$, for any two nodes $u, v \in V_i$, the number of neighbors of u in V_j and the number of neighbors of v in V_j are equal, this graph can also be termed *attributed regular graph*. Denote C_i as the attribute of nodes in V_i. Also, denote the number of neighbors in V_j of a node $v \in V_i$ as r_{ij}. Then, the configuration of this attributed regular graph can be represented as a set of tuples $\text{Config}(\mathcal{G}) = \{(C_i, C_j, r_{ij})\}_{i,j \in [k]}$.

Note that the definition of attributed regular graphs is similar to k-partite regular graphs, while attributed regular graphs allow edges connecting nodes from the same partition. It can be shown that the 1-WL test will color all the nodes of one partition in the same way. Based on the bound of representation power of MP-GNN (Theorem 5.2), we can obtain the following corollary about the impossibility of MP-GNN to distinguish GRL examples defined on attributed regular graphs. Fig. 5.11 gives some examples that illustrate the impossibility. Actually, with sufficient layers (iterations), MP-GNN (the 1-WL test) will always transform any attributed graph into

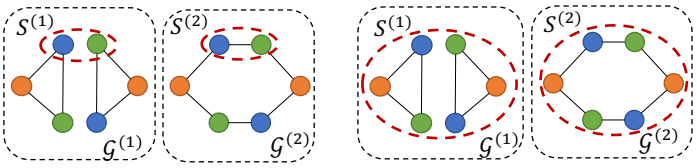


Fig. 5.11: A pair of attributed regular graphs $\mathcal{G}^{(1)}$, $\mathcal{G}^{(2)}$ with the same configuration and a proper selection of $S^{(1)}$, $S^{(2)}$: MP-GNN and the 1-WL test fail to distinguish $(\mathcal{G}^{(1)},S^{(1)})$, $(\mathcal{G}^{(2)},S^{(2)})$.

an attributed regular graph (Arvind et al, 2019) if we view the node representations obtained by MP-GNN as the node attributes on this transformed graph [1].

Corollary 5.1. *Consider two graph-structured features* $(\mathcal{G}^{(1)},S^{(1)})$, $(\mathcal{G}^{(2)},S^{(2)})$. *If two attributed regular graphs* $\mathcal{G}^{(1)}$, $\mathcal{G}^{(2)}$ *share the same configuration, i.e.,* $Config(\mathcal{G}^{(1)}) = Config(\mathcal{G}^{(2)})$, *and two multisets of attributes* $\{X_v^{(1)}|v \in S^{(1)}\}$ *and* $\{X_v^{(2)}|v \in S^{(2)}\}$ *are also equal, then* $f_{MP-GNN}(\mathcal{G}^{(1)},S^{(1)}) = f_{MP-GNN}(\mathcal{G}^{(2)},S^{(2)})$. *Therefore, if graph representation learning problems associate* $\{X_v^{(1)}|v \in S^{(1)}\}$ *and* $\{X_v^{(2)}|v \in S^{(2)}\}$ *with different targets, MP-GNN does not hold the expressive power to distinguish them and predict their correct targets.*

Proof. The proof is obtained by tracking each iteration of the 1-WL test and performing an induction.

Next, we will introduce several approaches that address the above limitations and that further improve the expressive power of MP-GNN .

5.4.2 Injecting Random Attributes

The main reason for limitations on the expressive power of MP-GNN is that MP-GNN does not track node identities; however, different nodes with the same attributes will be initialized with the same vector representations. This condition will be maintained unless their neighbors propagate different node representations. One way to improve the expressive power of MP-GNN is to inject each node with a unique attribute. Given a GRL example $(\mathcal{G},S)$, where $\mathcal{G} = (A,X)$,

$$g_I(\mathcal{G},S) = (\mathcal{G}_I,S), \quad \text{where } \mathcal{G}_I = (A,X \oplus \mathbf{I}), \tag{5.9}$$

where $\oplus$ is concatenation and $\mathbf{I}$ is an identity matrix, this gives each node a unique one-hot encoding and yields a new attributed graph G_I. The composite model

[1] Most transformed graphs have one single node per partition. In this case, two graphs that share the same configuration are isomorphic.

$f_{MP-GNN} \circ g_I$ increases expressive power as node identities are attached to the messages in the message passing framework and the distance and loop information can be learnt with sufficient iterations of message propagation.

However, the limitation of the above framework is that it is not permutation invariant (Def. 5.4): given that two isomorphic GRL examples $(\mathcal{G}^{(1)}, S^{(1)}) \cong (\mathcal{G}^{(2)}, S^{(2)})$, $g_I(\mathcal{G}^{(1)}, S^{(1)})$ and $g_I(\mathcal{G}^{(2)}, S^{(2)})$ may be not isomorphic any more. Then, the composite model $f_{MP-GNN} \circ g_I(\mathcal{G}^{(1)}, S^{(1)})$ may not equal $f_{MP-GNN} \circ g_I(\mathcal{G}^{(2)}, S^{(2)})$. As the obtained model loses the fundamental inductive bias of graph representation learning, it is hard to be generalized[2].

Remark 5.6. Some other approaches may share the same limitation with g_I, *e.g.,* using the adjacency matrix A (each row of A representing node attributes). However, Srinivasan and Ribeiro (2020a) argued that node embeddings obtained via matrix factorization, such as deepwalk (Perozzi et al, 2014) and node2vec (Grover and Leskovec, 2016), can keep the required invariance and thus are still generalizable. We will return to this concept in Sec. 5.4.2.4.

To overcome the above limitation, different methods have been proposed recently. These models share the following strategy: they first design some additional random node attributes Z, use them to argue the original dataset, and then learn a GNN model over the augmented dataset (Fig. 5.13).

The obtained models will be more expressive, as the random node attributes can be viewed as unique node identities that distinguish nodes. However, if the model is only trained based on a single GRL example augmented by these random attributes, it cannot keep invariance as discussed above. Instead, the model needs to be trained over multiple GRL examples augmented by independently injected random attributes. The new augmented GRL examples have the same target as the original GRL examples from which they are generated. This training of models over augmented examples essentially regularizes the permutation variance of the models and makes them behave almost "permutation invariant."

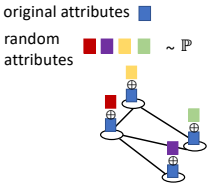
Different methods to inject these random attributes may be adopted, but a direct way is to attach Z to X, *i.e.,* given a graph-structured data $(\mathcal{G}, S)$, where $\mathcal{G} = (A, X)$,

$$g_Z(\mathcal{G}, S) = (\mathcal{G}_Z, S), \quad \text{where } \mathcal{G}_Z = (A, \tilde{X}_Z) \text{ and } \tilde{X}_Z \leftarrow X \oplus Z. \tag{5.10}$$

Note that for each realization Z, the composite model $f_{MP-GNN} \circ g_Z$ is not permutation invariant. Instead, all these approaches make $\mathbb{E}[f_{MP-GNN} \circ g_Z]$ permutation invariant and expect the models to keep invariant in expectation. To match such invariance in expectation, an approach must satisfy the following proposition.

Proposition 5.1. *The following two properties are needed to build a model by injecting random features Z.*

[2] Recent literature often states that the composite model is not inductive. Inductiveness and generalization to unobserved examples are related. In the transductive setting, $f_{MP-GNN} \circ g_I$ is less generalizable than f_{MP-GNN}, although the prediction performance of $f_{MP-GNN} \circ g_I$ may be sometimes better than f_{MP-GNN} due to the much stronger expressive power of $f_{MP-GNN} \circ g_I$.



Types of random attributes	Positional information	Model & reference
Random permutations	No	RP-GNN (Murphy et al, 2019)
(Almost uniform) Discrete r.v.	No	rGIN (Sato et al, 2020)
Distances to random anchor sets	Yes	PGNN (You et al, 2019)
Graph-convoluted Gaussian r.v.	Yes	CGNN (Srinivasan & Ribeiro, 2020)
Random signed Laplacian eigenmap	Yes	LE-GNN (Dwivedi et al, 2020)

Fig. 5.12: Injecting random node attributes can improve the expressive power of GNNs. Different types of random node attributes are adopted in different works. Some random node attributes contain node positional information (the position of a node with respect to other nodes in the graph).

1. *A sufficient number of Z's should be sampled during the training stage so that the model indeed captures permutation invariance in expectation.*
2. *The randomness in Z should be agnostic to the original node identities.*

To satisfy the property 1, a method suggests that for each forward pass to compute $f_{MP-GNN} \circ g_Z$ during the training stage, one Z should be re-sampled once or multiple times to get enough data argumentation. To satisfy the property 2, four different types of random Z have been proposed as described next.

5.4.2.1 Relational Pooling - GNN (RP-GNN) (Murphy et al, 2019a)

Murphy et al (2019a) considered randomly assigning an order of nodes as their extra attributes and proposed the model relational pooling GNN (RP-GNN). We use Z_{RP} to denote additional node attributes Z used in RP-GNN. Suppose the graph $\mathscr{G}$ has n nodes, Z_{RP} is uniformly sampled from all possible permutation matrices. That is, randomly pick a bijective mapping (permutation) $\pi : V(\mathscr{G}) \to V(\mathscr{G})$, and design permutation matrix $[Z_{RP}]_{ij} = 1$ if $j = \pi(i)$ and $[Z_{RP}]_{ij} = 0$ otherwise. Then, RP-GNN adopts the composite model,

$$f_{RP-GNN} = \mathbb{E}[f_{MP-GNN} \circ g_{Z_{RP}}]. \tag{5.11}$$

Theorem 5.4. *(Theorem 2.2 (Murphy et al, 2019a)) The RP-GNN f_{RP-GNN} is strictly more powerful than the original f_{MP-GNN}.*

Computing the expectation $\mathbb{E}[f_{MP-GNN} \circ g_{Z_{RP}}]$ is intractable as one needs to compute $f_{MP-GNN} \circ g_{Z_{RP}}$ for all possible permutations $\pi : V(\mathscr{G}) \to V(\mathscr{G})$. To overcome this problem, sampling of Z_{RP} may be needed.

However, as the entire permutation space is too large, uniformly random sampling of a limited number of Z_{RP} may introduce a large variance. To reduce the potential variance, Murphy et al (2019a) also proposed to sample all π's that permute only a small subset of nodes instead of the entire set of nodes. More recently, Chen et al (2020q) further adapted RP-GNN to solve the subgraph counting prob-

lem. They suggest to use all π's that permute all the nodes of each connected local subgraph.

5.4.2.2 Random Graph Isomorphic Network (rGIN) (Sato et al, 2021)

Sato et al (2021) generalized RP-GNN by setting the additional attributes of each node sampled from an almost uniform discrete probability distribution. The key difference from RP-GNN is that the additional attributes of two nodes are set to be independent of each other (while in RP-GNN, one-time random attributes of different nodes are correlated due to the nature of permutation). We use Z_r to denote Z used in rGIN and $[Z_r]_v$ to denote the attributes of node v. For example, they set

$$f_{rGIN} = \mathbb{E}[f_{MP-GNN} \circ g_{Z_r}], \text{ where } [Z_r]_v \sim \text{Unif}(\mathscr{D}) \, i.i.d. \, \forall \, v \in \mathscr{V}[\mathscr{G}],$$

where $\mathbb{E}$ indicates expectation and $\mathscr{D}$ is a discrete space with at least $1/p$ elements for some $p > 0$. Similar to RP-GNN, f_{rGIN} can be implemented by sampling only a few Z_r's for each evaluation of $f_{MP-GNN} \circ g_Z$ (indeed, one Z_r is sampled per forwarding evaluation (Sato et al, 2021)).

Theorem 5.5. *(Theorem 4.1 (Sato et al, 2021)) Consider a GRL example $(\mathscr{G}, v)$, where only a single node is contained in the node set of interest. For any graph-structured features $(\mathscr{G}', v')$, where the nodes of $\mathscr{G}'$ have a bounded maximal degree and the attributes X come from a finite space, then there exist an MP-GNN , such that:*

1. *If $(\mathscr{G}', v') \cong (\mathscr{G}, v)$, $f_{MP-GNN} \circ g_{Z_r}(\mathscr{G}', v') > 0.5$ with high probability.*
2. *If $(\mathscr{G}', v') \ncong (\mathscr{G}, v)$, $f_{MP-GNN} \circ g_{Z_r}(\mathscr{G}', v') < 0.5$ with high probability.*

This result can be viewed as a characterization of the expressive power of rGIN. However, this result is lessened by the fact that almost all nodes of all graphs will be associated with different representations within two iterations of the 1-WL test (so is MP-GNN) (Babai and Kucera, 1979). Moreover, the isomorphism problem of graphs with a bounded degree is known to be in P (Fortin, 1996). Instead, a very recent work was able to demonstrate the universal approximation of rGIN, which gives a stronger characterization of the expressive power of rGIN.

Theorem 5.6. *(Theorem 4.1 (Abboud et al, 2020)) Consider any invariant mapping $f^* : \mathscr{G}_n \to \mathbb{R}$, where $\mathscr{G}_n$ contains all graphs with n nodes. Then, there exists a rGIN $f_{MP-GNN} \circ g_{Z_r}$ such that*

$$p(|f_{MP-GNN} \circ g_{Z_r} - f^*| < \varepsilon) > 1 - \delta, \text{ for some given } \varepsilon > 0, \, \delta \in (0,1).$$

The above RP-GNN and rGIN adopt random attributes that are totally agnostic to the input data $(\mathscr{G}, S)$. Instead, the next two methods inject random attributes that leverage the input data. Particularly, these random attributes are related to the position/location of a node in the graph, which tends to counter the loss of positional information of nodes in MP-GNN.

5.4.2.3 Position-aware GNN (PGNN) (You et al, 2019)

You et al (2019) demonstrated that MP-GNN may not capture the position/location of a node in the graph, which is critical information for applications such as link prediction. Therefore, they proposed to use node positional embeddings as extra attributes. To capture permutation invariance in the sense of expectation, node positional embeddings are generated based on randomly selected anchor node sets. We denote the random attributes adopted in PGNN as Z_P, which is constructed as follows. Considering a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E}, X)$,

1. Randomly select a few anchor sets $(S_1, S_2, ..., S_K)$, where $S_k \subset \mathscr{V}$. Note that the choice of S_k is agnostic to the node identities: given a k, S_k will include each node with the same probability.
2. For some $u \in G$, set $[Z_P]_u = (d(u, S_1), ..., d(u, S_K))$ where $d(u, S_k)$, $k \in [K]$ is a distance metric between u and the anchor set S_k.

As the selection of the anchor sets is agnostic to node identities, the obtained Z_P still satisfies the property 2 in Proposition 5.1. Next, we specify the strategy to sample these anchor sets and the choice of the distance metric. The primary requirement to select those anchor sets is to keep low distortion of the two distances between nodes, where one distance is given by the original graph and the other one is given by those anchor sets. Specifically, distortion measures the faithfulness of the embeddings in preserving distances when mapping from one metric space to another metric space, which is defined as follows:

Definition 5.9. Given two metric spaces $(\mathscr{V}, d)$ and $(\mathscr{Z}, d')$ and a function $Z_P : \mathscr{V} \to \mathscr{Z}$, Z_P is said to have distortion α if $\forall u, v \in \mathscr{V}$, $\frac{1}{\alpha} d(u, v) \leq d'([Z_P]_u, [Z_P]_v) \leq d(u, v)$.

Fortunately, Bourgain (1985) showed the existence of a low distortion embedding that maps from any metric space to the l_p metric space:

Theorem 5.7. *(Bourgain's Theorem (Bourgain, 1985)) Given any finite metric space $(\mathscr{V}, d)$ with $|\mathscr{V}| = n$, there exists an embedding of $(\mathscr{V}, d)$ into $\mathbb{R}^K$ under any l_p metric, where $K = O(\log^2 n)$, and the distortion of the embedding is $O(\log n)$.*

Based on a constructive proof of Theorem 5.7, Linial et al (1995) provide an algorithm to construct an $O(\log^2 n)$ dimensional embedding via anchor sets. This yields the selection of anchor sets and the definition of the distance metric to define Z_P, which are adopted by PGNN (You et al, 2019).

By selecting $K = c \log^2 n$, many random sets $S_{i,j} \subset \mathscr{V}, i = 1, 2, ..., \log n, j = 1, 2, ..., c \log n$, where c is a constant, $S_{i,j}$ is chosen by including each point in $\mathscr{V}$ independently with probability $\frac{1}{2^i}$. We further define

$$[Z_P]_v = \left(\frac{d(v, S_{1,1})}{k}, \frac{d(v, S_{1,2})}{k}, ..., \frac{d(v, S_{\log n, c \log n})}{k}\right) \tag{5.12}$$

where $d(v, S_{i,j}) = \min_{u \in S_{i,j}} d(v, u)$. Then, Z_P is an embedding method that satisfies Theorem 5.7.

Compared with RP-GNN and rGIN, the random attributes adopted by PGNN deal specifically with the positional information of a node in graph. Therefore, PGNN is better for the tasks that are directly related to the positions of nodes, *e.g.,* link prediction. You et al (2019) did not provide a mathematical characterization of the representation power of PGNN. However, the way to establish Z_P allows that for the two nodes u, v, the attributes $[Z_P]_u$ and $[Z_P]_v$ by definition are statistically correlated. As for the example in Fig. 5.9, such correlation gives PGNN the information that the distance between Lynx and Pelagic Fish is different from the distance between Orca and Pelagic Fish, and thus may successfully distinguish $(G, \{\text{Lynx}, \text{Pelagic Fish}\})$ and $(G, \{\text{Orca}, \text{Pelagic Fish}\})$ and making the right link prediction.

Note that the original PGNN (You et al, 2019) does not use MP-GNN as the backbone to perform message passing. Instead, PGNN allows message passing from nodes to anchor sets. As such, this approach is not directly relevant to the expressive power and is thus out of the scope of this chapter, so we will not discuss it in detail. Interested readers may refer to the original paper (You et al, 2019).

5.4.2.4 Randomized Matrix Factorization (Srinivasan and Ribeiro, 2020a)(Dwivedi et al, 2020)

Srinivasan and Ribeiro (2020a) recently made an important observation that node positional embeddings obtained via the factorization of some variants of the adjacency matrix A can be used as node attributes as long as certain random perturbation is allowed. The obtained models still keep permutation invariance in expectation. Srinivasan and Ribeiro (2020a) argue that a model that is built upon these random perturbed node positional embeddings is still inductive and holds good generalization properties. This significant observation challenges the traditional claim that models built upon these node positional embeddings are not inductive. A high-level idea of why this is true is as follows: suppose the SVD decomposition of the adjacency matrix $A = U\Sigma U^T$. When we permute the order of nodes, that is, the row and column orders of A, the row order of U will be changed simultaneously. Therefore, the models that use U as the node attributes should keep the permutation invariance. That randomly perturbed factorization is needed because such SVD decomposition is not unique.

Although Srinivasan and Ribeiro (2020a) proposed this idea, they did not explicitly compute the node positional embeddings via matrix factorization. Instead, their method samples a series of Gaussian random matrices $Z_{G,1}, Z_{G,2}, \ldots$ and let them propagate over the graph, *e.g.,* for the two hops,

$$Z_G = \psi(\hat{A}\,\psi(\hat{A}\,Z_{G,1}) + Z_{G,2}),$$

where ψ's are MLPs and $\hat{A}$ indicates some variant of the adjacency matrix. The rows of Z_G essentially give rough node positional embeddings. Then, these obtained node embeddings are further used as the attributes of nodes in MP-GNN.

Dwivedi et al (2020) indeed adopted matrix factorization explicitly. They proposed to use the randomly perturbed Laplacian eigenmaps as the additional attributes. Specifically, suppose the normalized Laplacian matrix is defined as

$$L = I - D^{-1/2}AD^{-1/2},$$

where D is the diagonal degree matrix. Denote the eigenvalue decomposition of L as $L = U\Sigma U^T$. The eigenvalue decomposition is not unique, so we assume that U can be arbitrarily chosen from all the potential choices. Fortunately, if there are no multiple eigenvalues, this U is unique for each column up to a $\pm$ sign. Then, we may directly set the extra node attributes as

$$Z_{LE} = U\Gamma, \text{ where } \Gamma_{ii} \sim \text{Unif}(\{-1,1\}) \; i.i.d. \; \forall i \in [|V|], \; \Gamma_{ij} = 0, \; \forall i \neq j, \quad (5.13)$$

where Γ is a diagonal matrix where diagonal elements are uniformly independently set as 1 or -1. Here, U can be replaced with a few slices of the columns of U. Let $g_{Z_{LE}}$ denote the operation to concatenate these additional attributes Z_{LE} with the original node attributes. Then, the overall composite model becomes $f_{MP-GNN} \circ g_{Z_{LE}}$. The following lemma shows that the permutation invariance of $f_{MP-GNN} \circ g_{Z_{LE}}$ in expectation if the Laplacian matrices hold distinct eigenvalues:

Lemma 5.3. *If $(\mathcal{G}^{(1)}, S^{(1)}) \cong (\mathcal{G}^{(2)}, S^{(2)})$ and if there are no multiple eigenvalues of their corresponding normalized Laplacian matrices, then any choice of eigenvalue decomposition to obtain node embeddings will yield*

$$\mathbb{E}(f_{MP-GNN} \circ g_{Z_{LE}}(\mathcal{G}^{(1)}, S^{(1)})) = \mathbb{E}(f_{MP-GNN} \circ g_{Z_{LE}}(\mathcal{G}^{(2)}, S^{(2)})).$$

Proof. The proof can be easily seen from the above arguments.

As shown in Lemma 5.3, the composite model keeps permutation invariance in expectation for most graphs, although it may break invariance in some corner cases. Regarding the expressive power, Z_{LE} associates different nodes with distinct attributes because U is an orthogonal matrix by definition. Hence, there must exist $f_{MP-GNN} \circ g_{Z_{LE}}$ that may distinguish any node subsets from the graph:

Theorem 5.8. *For any two GRL examples $(\mathcal{G}, S^{(1)})$, $(\mathcal{G}, S^{(2)})$ over the same graph $\mathcal{G}$, even if they are isomorphic, as long as $S^{(1)} \neq S^{(2)}$, there exists an f_{MP-GNN} such that $f_{MP-GNN} \circ g_{Z_{LE}}(\mathcal{G}, S^{(1)}) \neq f_{MP-GNN} \circ g_{Z_{LE}}(\mathcal{G}, S^{(2)})$. However, if those two GRL examples are indeed isomorphic $(\mathcal{G}, S^{(1)}) \cong (\mathcal{G}, S^{(2)})$ over the same graph $\mathcal{G}$ and the normalized Laplacian matrix of $\mathcal{G}$ has no multiple same-valued eigenvalues, then $\mathbb{E}(f_{MP-GNN} \circ g_{Z_{LE}}(\mathcal{G}, S^{(1)})) = \mathbb{E}(f_{MP-GNN} \circ g_{Z_{LE}}(\mathcal{G}, S^{(2)}))$.*

Proof. The proof can be easily seen from the above arguments.

Theorem 5.8 implies the potential of $f_{MP-GNN} \circ g_{Z_{LE}}$ to distinguish different node sets from the same graph. Note that although $f_{MP-GNN} \circ g_{Z_{LE}}$ achieves great representation power, it does not always work very well for link prediction in practice (Dwivedi et al, 2020) when compared with another model SEAL (Zhang and

Chen, 2018b) (compare their performance on the COLLAB dataset in (Hu et al, 2020b)). SEAL is based on the deterministic distance attributes that are introduced in the next subsection. Whether a model is permutation invariant is a much weaker statement on characterizing the generalization of the model. Actually, when the model is paired node positional embeddings, the dimension of the parameter space increases, and thus also negatively impacts the generalization. A comprehensive investigation of this observation is left for future study.

In the next subsection, we will introduce deterministic node distance attributes, which provide a different angle to solve the above problem. Distance encoding has a solid mathematical foundation and provides the theoretical support for many empirically well-behaved models such as SEAL (Zhang and Chen, 2018b) and ID-GNN (You et al, 2021).

5.4.3 Injecting Deterministic Distance Attributes

In this subsection, we will introduce an approach that boosts the expressive power of MP-GNN by injecting deterministic distance attributes.

The key motivation behind the deterministic distance attributes is as follows. In Section 5.4.1, we have shown that MP-GNN is limited in its ability to measure the distances between different nodes, to count cycles[3], and to distinguish attributed regular graphs. All of these limitations are essentially inherited from the 1-WL test which does not capture distance information between the nodes. If MP-GNN is paired with some distance information, then the composite model must achieve more expressive power. Then, the question is how to inject the distance information properly.

There are two important pieces of intuition to design such distance attributes. First, the effective distance information is typically correlated with the tasks. For example, consider a GRL example $(\mathcal{G}, S)$. If this task is node classification ($|S| = 1$), the information of distance from this node to itself (thus the cycles containing this node) is relevant because it measures the information of the contextual structure. If the task is link prediction ($|S| = 2$), the information of distance between the two end nodes of the link is relevant as two nodes near to each other in the network tend to be connected by a link. For graph-level prediction ($S = \mathcal{V}(\mathcal{G})$), the information of distances between any pairs of nodes could be relevant as it can be viewed as a group of link predictions. Second, besides the distance between the nodes in S, the distance from S to other nodes in G may also provide useful side-information. Both two aspects inspire the design of distance attributes.

There have been a few empirically successful GNN models that leverage deterministic distance attributes, although their impact on the expressive power of GNNs

[3] Cycles actually carry a special type of distance information, as they describe the length of walks from one node to itself. If the distance from one node to itself is not measured by the shortest path distance but by the returning probability of random walk, this distance already contains the cycle information.

has not been characterized until very recently (Li et al, 2020e). For link prediction, Li et al (2016a) first consider annotating the two end-nodes of the link of interest. These two end-nodes are annotated with one-hot encodings and all other nodes are annotated by zeros. Such annotations can be transformed into distance information via GNN message passing. Again for link prediction, Zhang and Chen (2018b) first sample an enclosing subgraph around the queried link and then annotate each node in this subgraph with one-hot encodings of the shortest path distances (SPDs) from this node to the two end-nodes of the link. Note that deciding whether a node is in the enclosing subgraph around the queries link already gives a distance attribute. Zhang and Chen (2019) uses a similar idea to perform matrix completion which is a similar task to link prediction. For graph classification and graph-level property prediction, Chen et al (2019a) and Maziarka et al (2020a) adopt the SPDs between two nodes as edge attributes. These edge attributes can be also used as the input of MSG (Eq.equation 11.45) in MP-GNN. You et al (2021) annotates a node as 1 and other nodes as 0 to improve MP-GNN in node classification. As our focus is on the theoretical characterization of the expressive power, we will not go into detail about these empirically successful works. Interested readers are referred to the relevant papers.

Remark 5.7. (Comparison between deterministic distance attributes and random attributes) Deterministic distance attributes have some advantages. First, as there is no randomness in the input attributes, the optimization procedure of the model contains less noise. Hence, the training procedure tends to converge much faster than the model with random attributes. The model evaluation performance contains much less noise too. Some empirical evaluation of the convergence of the model training with random attributes can be found in Abboud et al (2020). Second, a model based on deterministic distance attributes typically shows better generalization in practice than the one based on random attributes. Although theoretically a model is permutation invariant when being trained based on sufficiently many examples with random attributes (as discussed in Sec.5.4.2), in practice, this could be hard to achieved due to the high complexity.

Deterministic distance attributes have some disadvantages. First, models that are paired with deterministic attributes may never achieve the universal approximation, unless the graph isomorphism problem is in P. However, random attributes may be universal in the probabilistic sense (*e.g.,* Theorem 5.6). Second, deterministic distance attributes typically depend on the information S in a GRL example $(\mathcal{G}, S)$. This introduces an issue in computation: that is, if there are two GRL examples $(\mathcal{G}^{(1)}, S^{(1)})$ and $(\mathcal{G}^{(2)}, S^{(2)})$ sharing the same graph $\mathcal{G}$ but with different node sets of interest $S^{(1)} \neq S^{(2)}$, they will be attached with different deterministic distance attributes and hence GNNs have to make inference over them separately. However, GNNs with random attributes can share intermediate node representations $\{\mathbf{h}_v^{(L)} | v \in \mathcal{V}[\mathcal{G}]\}$ in Eq.equation 5.4, between the two GRL examples, which saves intermediate computation.

5.4.3.1 Distance Encoding (Li et al, 2020e)

Suppose we aim to make prediction for a GRL example $(\mathcal{G}, S)$. Li et al (2020e) defined distance encoding $\zeta(u|S)$ as an extra node attribute for node $u \in \mathcal{V}[\mathcal{G}]$.

Definition 5.10. For a GRL example $(\mathcal{G}, S)$ where $\mathcal{G} = (A, X)$. *Distance encoding* $\zeta(u|S)$ for node u is defined as follows

$$\zeta(u|S) = \sum_{v \in S} \text{MLP}(\zeta(u|v)) \tag{5.14}$$

where $\zeta(u|v)$ charaterizes a certain distance between u and v. We may choose

$$\zeta(u|v) = \text{g}(\ell_{uv}),\ \ell_{uv} = (1, (W)_{uv}, (W^2)_{uv}, ..., (W^k)_{uv}, ...), \tag{5.15}$$

where $W = AD^{-1}$ is the random walk matrix and $g(\cdot)$ is a general function that maps ℓ_{uv} to different types of distance measures.

Note that $\zeta(u|S)$ depends on the graph structure $\mathcal{G}$, which we omit in our notation for simplicity. First, setting $g(\ell_{uv})$ as the first non-zero position in ℓ_{uv} gives the *shortest-path-distance (SPD)* from v to u. Second, setting $g(\ell_{uv})$ as follows gives *generalized PageRank scores* (Li et al, 2019f):

$$\zeta_{gpr}(u|v) = \sum_{k \geq 1} \gamma_k (W^k)_{uv} = (\sum_{k \geq 0} \gamma_k W^k)_{uv}, \quad \gamma_k \in \mathbb{R}, \text{ for all } k \in \mathbb{Z}_{\geq 0}\ . \tag{5.16}$$

Different choices of $\{\gamma_k | k \in \mathbb{Z}_{\geq 0}\}$ yield various distance measures between u and v.

Personalized PageRank scores (Jeh and Widom, 2003): $\gamma_k = \alpha^k$, $\alpha \in (0,1)$,

Heat-kernel PageRank scores (Chung, 2007): $\gamma_k = \beta^k e^{-\beta}/k!$, $\beta > 0$,

Inverse hitting time (Lovász et al, 1993): $\gamma_k = k$.

It is important to see that the above definition of distance encoding satisfies permutation invariance.

Lemma 5.4. *For two isomorphic GRL examples* $(\mathcal{G}^{(1)}, S^{(1)}) \overset{\pi}{\cong} (\mathcal{G}^{(2)}, S^{(2)})$*, if* $\pi(u) = \pi(v)$ *for* $u \in \mathcal{V}[\mathcal{G}^{(1)}]$ *and* $v \in \mathcal{V}[\mathcal{G}^{(2)}]$*, their distance encodings are equal* $\zeta(u|S^{(1)}) = \zeta(v|S^{(2)})$.

Proof. The proof can be easily seen by the definition of distance encoding.

Li et al (2020e) considers using distance encoding as node extra attributes. Specifically, MP-GNN can be improved by setting $\tilde{X}_v = X_v \oplus \zeta(v|S)$,where $\oplus$ is the concatenation. The obtained model is termed DE-GNN, denoted as f_{DE}.

DE-GNN has been shown to be more powerful than MP-GNN. Recall that the fundamental limit of MP-GNN is the 1-WL test for graph representation learning problems (Theorem 5.2). Corollary 5.1 further indicates that attributed regular graphs may not be distinguished by MP-GNN under certain scenarios. Li et al

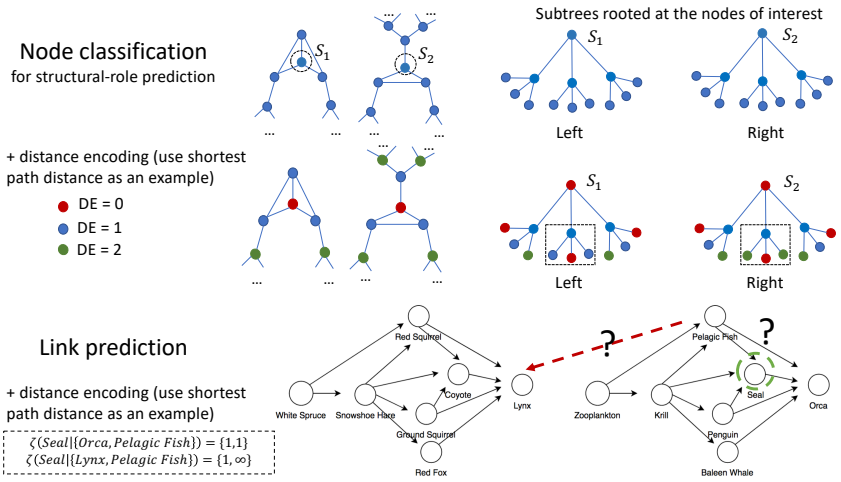


Fig. 5.13: Distance encoding can be used to distinguish non-isomorphic graph-structured examples. In the example of node classification, we consider classifying nodes based on their roles in their contextual structures, termed structural roles (Henderson et al, 2012). Nodes in S_1 and S_2 have different structure roles. However MP-GNN with two layers will confuse these two nodes; while with distance encoding, DE-GNN may distinguish them. In the example of link prediction, although two nodes $\{\text{Lynx}, G\}$ and $\{\text{Orca}, G\}$ are isomorphic (where we ignore the node identities), distance encoding on the node Seal allows us to distinguish node pairs $\{\text{Orca, Pegagic Fish}\}$ and $\{\text{Lynx, Pegagic Fish}\}$.

(2020e) considers the scenario when the graphs are regular and do not have attributes and proved that DE-GNN can distinguish two GRL examples with high probability, which is formally stated in the following theorem.

Theorem 5.9. *(Theorem 3.3 (Li et al, 2020e)) Consider two GRL examples $(\mathcal{G}^{(1)}, S^{(1)})$ and $(\mathcal{G}^{(2)}, S^{(2)})$ where $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$ are two n-sized unattributed regular graphs, and $|S^{(1)}| = |S^{(2)}|$ is a constant (independent of n). Suppose $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$ are uniformly independently sampled from all n-sized r-regular graphs where $3 \leq r < (2\log n)^{1/2}$. Then, for any small constant $\varepsilon > 0$, there exists DE-GNN with certain weights within $L \leq \lceil (\frac{1}{2} + \varepsilon) \frac{\log n}{\log(r-1)} \rceil$ layers that can distinguish these two examples with high probability. Specifically, the outputs $f_{DE}((\mathcal{G}^{(1)}, S^{(1)})) \neq f_{DE}((\mathcal{G}^{(2)}, S^{(2)}))$ with probability $1 - o(n^{-1})$. The specific form of DE, i.e., g in Eq.equation 5.15, can be simply chosen as short path distance. The little-o notation here and later are w.r.t. n.*

Theorem 5.9 focuses on the node sets of unattributed regular graphs. We conjecture that the statement can be generalized to attributed regular graphs as distinct attributes can only further improve the distinguishing power of a model. Moreover,

the assumption on regularity of graphs is also not crucial because the 1-WL test or MP-GNN may transform all graphs, attributed or not, into attributed regular graphs with enough iterations (Arvind et al, 2019).

Of course, DE-GNN may not distinguish any non-isomorphic GRL examples. Li et al (2020e) introduce the limitation of DE-GNN. Particularly, DE-GNN cannot distinguish nodes of distance regular graphs with the same intersection arrays, although DE-GNN may distinguish their edges (See Fig. 5.14 later). Li et al (2020e) also generalize the above results to the case that leverages distance attributes as edge attributes (to control message aggregation in MP-GNN). Interested readers can check the details in their original paper.

5.4.3.2 Identity-aware GNN (You et al, 2021)

As a concurrent work with DE-GNN, You et al (2021) studied a special type of distance encoding to improve the node representations learnt by MP-GNN. Specifically, when MP-GNN is adopted to compute the representation of node v, You et al (2021) suggests attaching each node u in the graph with an extra binary attribute $\zeta_{ID}(u|\{v\})$ to indicate the identity of node v where

$$\zeta_{ID}(u|\{v\}) = \begin{cases} 1 & \text{if } u = v, \\ 0 & \text{o.w.} \end{cases} \tag{5.17}$$

MP-GNN that leverages $\zeta_{ID}(u|\{v\})$ is termed Identity-aware GNN (ID-GNN). $\zeta_{ID}(u|\{v\})$ is a simple implementation of distance encoding (Eq. equation 5.14) when the set S contains only one node v. Although ID-GNN does not compute distance measures as DE-GNN, ID-GNN holds the same representation power as DE-GNN for node classification, as the distance information from another node u to the target node v can be learnt by ID-GNN via an extra identity attribute.

Theorem 5.10. *For two graph-structured examples $(\mathcal{G}^{(1)}, S^{(1)})$ and $(\mathcal{G}^{(2)}, S^{(2)})$, where $|S^{(i)}| = 1$ for $i \in \{1,2\}$ and $\mathcal{G}^{(i)}$ is unattributed, if DE-GNN can distinguish them with L layers, then ID-GNN requires at most $2L$ layers to distinguish them.*

Proof. ID-GNN needs the first L layers to propagate the identity attribute to capture distance information and the second L layers to let such information propagate back to finally be merged into the node representations.

Although ID-GNN adopts a specific type of DE to learn node representations, ID-GNN was also used to perform graph-level prediction (You et al, 2021). Specifically, for every node v in the graph G, ID-GNN attaches 1 to this node, 0's to other nodes and computes the node representation h_v. Iterating over all the nodes, ID-GNN collects all node representations $\{\mathbf{h}_v | v \in \mathcal{V}(\mathcal{G})\}$. Then, by following Eq.equation 5.4 (S is the entire node set $\mathcal{V}(\mathcal{G})$ here), ID-GNN can aggregate the node representations of all the nodes and further make graph-level predictions. Actually, combining the statement of Theorem 5.9 and the union bound, Li et al (2020e) indicates the

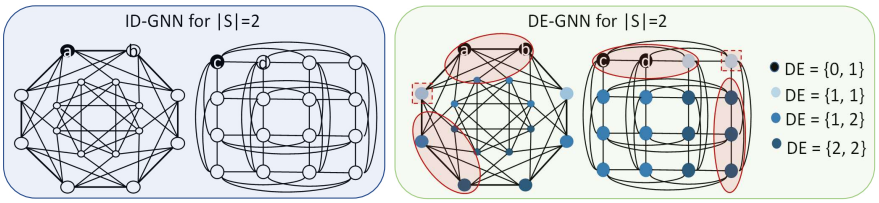


Fig. 5.14: ID-GNN *v.s.* DE-GNN makes predictions for a pair of nodes. Two graphs are the Shrikhande graph (left) and the 4×4 Rook's graph (right). ID-GNN (black nodes attached identities) cannot distinguish node pairs $\{a,b\}$ and $\{c,d\}$. DE-GNN may learn distinct representations of $\{a,b\}$ and $\{c,d\}$. In these two graphs, each node is colored with its DE that is a set of SPDs to either node in the target node sets $\{a,b\}$ or $\{c,d\}$ (Eq. equation 5.14). Note that the neighbors of nodes with DE$=\{1,1\}$ (dashed boxes) are enclosed by red ellipses which shows that the neighbors of these two nodes have different DE's. Hence, after one layer of DE-GNN, the intermediate representations of nodes with DE$=\{1,1\}$ are different between these two graphs. Using another layer, DE-GNN can distinguish the representations of $\{a,b\}$ and $\{c,d\}$.

expressive power of the above procedure for the entire graph classification problem, summarized in the following corollary.

Corollary 5.2. *(Reproduced from Corollary 3.4 (Li et al, 2020e)) Consider two GRL examples $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$. Suppose $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$ are uniformly independently sampled from all n-sized unattributed r-regular graphs where $3 \leq r < (2\log n)^{1/2}$. Then, ID-GNN with a sufficient number of layers can distinguish these two graphs with probability $1 - o(1)$. The little-o notation here and later are w.r.t. n.*

ID-GNN can be viewed as the simplest version of DE-GNN that achieves the same expressive power for node-level prediction. However, when the prediction tasks contain two nodes, *i.e.,* node-pair-level prediction, ID-GNN will be less powerful than DE-GNN.

To make a prediction for a GRL example $(\mathcal{G}, S)$ where $|S| = 2$, ID-GNN can adopt two different approaches. First, ID-GNN can attach the extra identity attributes to the two nodes in S separately, learn their representations separately and combine these two representations to make the final prediction. However, this approach cannot capture the distance information between the two nodes in S. Instead, ID-GNN uses an alternative approach. ID-GNN attaches the extra identity attribute to only one of nodes in S and performs message passing. Then, after a sufficient number of layers where the extra node identity is propagated from one node to another in S, the distance information between these two nodes can be captured. Finally, ID-GNN makes its prediction based on the two node representations in S. Note that although the second approach captures the distance information between the two nodes in S, it is still less powerful than DE-GNN. One example is shown in Fig. 5.14.

Up to this point, we have mostly focused on the message passing framework of GNNs, which leverages the sparsity of real-world graphs. In the next subsection, we remove the need for sparsity and discuss higher-order GNNs. These GNNs essentially mimic higher-dimensional WL tests and achieve more expressive power.

5.4.4 Higher-order Graph Neural Networks

The final collection of techniques for building GNNs, which overcome the limitation of the 1-WL test, are related to higher-dim WL test. In this subsection, for notational simplicity, we focus only on graph-level prediction learning problems, where higher-order GNNs are mostly used.

The family of WL tests forms a hierarchy for the graph isomorphism problem (Cai et al, 1992). There are different definitions of the higher-dim WL tests. We follow the terminology adopted in Maron et al (2019a) and introduce two types of WL tests: the k-forklore WL (k-FWL) test and the k-WL test.

Recall $\mathcal{G}^{(i)} = \{A^{(i)}, X^{(i)}\}$, $i \in \{1,2\}$. For both $\mathcal{G}^{(i)}$'s, $i \in \{1,2\}$, do the following steps.

1. For each k-tuple of node set $V_j = (v_{j_1}, v_{j_2}, ..., v_{j_k}) \in \mathcal{V}^k$, $j \in [n]^k$, we initialize V_j with a color denoted by $C_j^{(0)}$. These colors satisfy the condition that for two k-tuples, say V_j and $V_{j'}$, $C_j^{(0)}$ and $C_{j'}^{(0)}$ are the same if and only if: (1) $X_{v_{j_a}} = X_{v_{j'_a}}$; (2) $v_{j_a} = v_{j_b} \Leftrightarrow v_{j'_a} = v_{j'_b}$; and (3) $(v_{j_a}, v_{j_b}) \in \mathcal{E} \Leftrightarrow (v_{j'_a}, v_{j'_b}) \in \mathcal{E}$ for all $a, b \in [k]$.
2. **k-FWL:** For each k-tuple V_j and $u \in V$, define $N_{k-FWL}(V_j; u)$ as a k-tuple of k-tuples, such that $N_{k-FWL}(V_j; u) = ((u, v_{j_2}, ..., v_{j_k}), (v_{j_1}, u, ..., v_{j_k}), (v_{j_1}, v_{j_2}, ..., u))$. Then the color of V_i can be updated via the following mapping.

$$\textbf{Update colors:} \quad C_j^{(l+1)} \leftarrow \text{HASH}(C_j^{(l)}, \{(C_{j'}^{(l)} | V_{j'} \in N_{k-FWL}(V_j; u))\}_{u \in V}). \tag{5.18}$$

   k-WL: For each k-tuple V_j and $u \in \mathcal{V}$, define $N_{k-WL}(V_j; u)$ as a set of k-tuples such that $N_{k-WL}(V_j; u) = \{(u, v_{j_2}, ..., v_{j_k}), (v_{j_1}, u, ..., v_{j_k}), (v_{j_1}, v_{j_2}, ..., u)\}$ Then, the color of V_i can be updated via the following mapping.

$$\textbf{Update colors:} \quad C_j^{(l+1)} \leftarrow \text{HASH}(C_j^{(l)}, \cup_{u \in V} \{C_{j'}^{(l)} | V'_j \in N_{k-WL}(V_j; u)\}), \tag{5.19}$$

   where the HASH operations in both cases guarantee an injective mapping with different inputs yielding different outputs.
3. For each step l, $\{C_j^{(l)}\}_{j \in [V(G^{(i)})]^k}$ is a multi-set. If such multi-sets of the two graphs are not equal, return $\mathcal{G}^{(1)} \ncong \mathcal{G}^{(2)}$. Otherwise, go to Eq. equation 5.19.

Similar to the 1-WL test, if the k-(F)WL test returns $\mathcal{G}^{(1)} \ncong \mathcal{G}^{(2)}$, then it follows that $\mathcal{G}^{(1)}$, $\mathcal{G}^{(2)}$ are not isomorphic. However, the reverse is not true.

Fig. 5.15: Use k-FWL and k-WL to distinguish $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$.

The key idea of these higher-dim WL tests is to color every k-tuple of nodes in the graphs and update these colors by aggregating the colors from other k-tuples that

share $k-1$ nodes. The procedures of the k-FWL test and the k-WL test are shown in Fig. 5.15. Note that they perform aggregation differently, and as such, have different power to distinguish non-isomorphic graphs. These two types of tests form a nested hierarchy, as summarized in the following theorem.

Theorem 5.11. *(Cai et al, 1992; Grohe and Otto, 2015; Grohe, 2017)*

1. *The k-FWL test and the $k+1$-WL test have the same discriminatory power, for $k \geq 1$.*
2. *The 1-FWL test, the 2-WL test and the 1-WL test have the same discriminatory power.*
3. *There are some graphs that the $k+1$-WL test can distinguish while the k-WL test cannot, for $k \geq 2$.*

Because of Theorem 5.11, GNNs that are able to capture the power of these higher-dim WL tests can be strictly more powerful than the 1-WL test. Therefore, higher-order GNNs have the potential to learn even more complex functions than MP-GNN.

However, the drawback of these GNNs is their computational complexity. By the definition of higher-order WL tests, the colors of all k-tuples of nodes need to be tracked. Correspondingly, higher-order GNNs that mimic higher-order WL tests need to associate each k-tuple with a vector representation. Therefore, their memory complexity is at least $\Omega(|\mathcal{V}|^k)$, where $|\mathcal{V}|$ is the number of nodes in the graph. The computational complexity is at least $\Omega(|\mathcal{V}|^{k+1})$, which makes these higher-order GNNs prohibitively expensive for large-scaled graphs.

5.4.4.1 k-WL-induced GNNs (Morris et al, 2019)

Morris et al (2019) first proposed k-GNN by following the k-WL test. Specifically, k-GNN associates each k-tuple of nodes, denoted by V_j, $j \in \mathcal{V}^k$, with a vector representation that is initialized as $\mathbf{h}_j^{(0)}$. In order to save memory, k-GNN only considers k-tuples that contain k different nodes and ignores the order of these nodes. Therefore, each k-tuple reduces to a set of k nodes. With some modification of notation in this subsection, let $\mathcal{V}_j$ denote this set of k different nodes. The initial representation of $\mathcal{V}_j$, $\mathbf{h}_j^{(0)}$ is chosen as a one-hot encoding such that $\mathbf{h}_j^{(0)} = \mathbf{h}_{j'}^{(0)}$, if and only if the subgraphs induced by V_j and $V_{j'}$ are isomorphic.

Then, k-GNN follows the following update procedure of these representations:

$$\mathbf{h}_j^{(l+1)} = \text{MLP}(\mathbf{h}_j^{(l)} \oplus \sum_{V_{j'}:N_{k-GNN}(V_j)} \mathbf{h}_{j'}^{(l)}), \quad \forall\, k\text{-sized node sets } V_j, \tag{5.20}$$

where $N_{k-GNN}(V_j) = \{V_{j'} \,|\, |V_{j'} \cap V_j| = k-1\}$. Note that $N_{k-GNN}(V_j)$ defines the neighbors of V_j differently than N_{k-WL} (see Eq.equation 5.19), because V_j is now a k-sized node set instead of a k-tuple.

Eq.equation 5.20 has time complexity at least $O(|\mathcal{V}|^k)$ as the size of $N_{k-GNN}(V_j)$ is $O(|\mathcal{V}|^k)$. Recently, Morris et al (2019) also considers using a local neighborhood of V_j instead of $N_{k-GNN}(V_j)$. This local neighborhood only includes $V_{j'} \in N_{k-GNN}(V_j)$, such that the node in $V_{j'} \backslash V_j$ is connected to at least one node in V_j. Morris et al (2020b) demonstrated that a variant of this local version of k-GNN may be as powerful as the k-WL test, although a deeper architecture with more layers is needed to match the expressive power.

k-GNN is at most as powerful as the k-WL test. To be more expressive than MP-GNN, $k = 3$ is needed. Therefore, the memory complexity is at least $\Omega(|\mathcal{V}|^3)$. Subsequently, the computational complexity of k-GNN, even for their local version, is at least $\Omega(|\mathcal{V}|^3)$ per layer.

5.4.4.2 Invariant and equivariant GNNs (Maron et al, 2018, 2019b)

To build higher-order GNNs, every k-tuple needs to be associated with a vector representation. Therefore, regardless whether a local or a global neighborhood aggregation is adopted (Eq.equation 5.20), the benefit of reducing the computation by leveraging the sparse graph structure is limited, as it cannot reduce the dominant term $\Omega(|\mathcal{V}|^k)$. Moreover, to handle a sparse graph structure, these higher-order GNNs also need random memory access, which introduces additional computational overhead. Therefore, a line of research into building higher-order GNNs totally ignores graph sparsity. Graphs are viewed as tensors and NNs take these tensors as input. The NNs are designed to be invariant to the order of tensor indices.

Many approaches (Maron et al, 2018, 2019a,b; Chen et al, 2019f; Keriven and Peyré, 2019; Vignac et al, 2020a; Azizian and Lelarge, 2020) adopt this formulation to build GNNs and analyze their expressive power.

Each k-tuple $V_j \in V^k$ is associated with a vector representation $\mathbf{h}_j^{(l)}$. We assume that $\mathbf{h}_j^{(l)} \in \mathbb{R}$ for simplicity. By concatenating the k-tuple's representations together, we obtain a k-order tensor:

$$H \in \mathbb{R}^{\otimes_k |\mathcal{V}|}, \quad \text{where} \quad \mathbb{R}^{\otimes_k |\mathcal{V}|} = \mathbb{R}^{\underbrace{|\mathcal{V}| \times \cdots \times |\mathcal{V}|}_{k \text{ times}}}.$$

Maron et al (2018) investigates linear permutation invariant and equivariant mappings defined on $\mathbb{R}^{\otimes_k |\mathcal{V}|}$.

Definition 5.11. Given a bijective mapping $\pi : \mathcal{V} \to \mathcal{V}$ and $H \in \mathbb{R}^{\otimes_k |\mathcal{V}|}$, define $\pi(H) := H'$, where $H'_{(\pi(v_1),\pi(v_2),\ldots,\pi(v_k))} = H_{(v_1,v_2,\ldots,v_k)}$, for all k-tuples $(v_1, v_2, \ldots, v_k) \in \mathcal{V}^k$.

Definition 5.12. A mapping $g : \mathbb{R}^{\otimes_k |\mathcal{V}|} \to \mathbb{R}$ is called *invariant*, if for any bijective mapping $\pi : \mathcal{V} \to \mathcal{V}$ and $H \in \mathbb{R}^{\otimes_k |\mathcal{V}|}$, $g(H) = g(\pi(H))$.

Definition 5.13. A mapping $g : \mathbb{R}^{\otimes_k |\mathcal{V}|} \to \mathbb{R}^{\otimes_k |\mathcal{V}|}$ is called *equivariant*, if for any bijective mapping $\pi : \mathcal{V} \to \mathcal{V}$ and $H \in \mathbb{R}^{\otimes_k |\mathcal{V}|}$, $\pi(g(H)) = g(\pi(H))$.

Maron et al (2018) showed that the number of the bases needed to represent all possible linear invariant mappings from $\mathbb{R}^{\otimes_k|\mathcal{V}|} \to \mathbb{R}$ is $b(k)$, where $b(k)$ is the k-th Bell number. Additionally, the number of bases, needed to represent all possible linear equivariant mappings from $\mathbb{R}^{\otimes_k|\mathcal{V}|} \to \mathbb{R}^{\otimes_{k'}|V|}$, is $b(k+k')$. To better understand this observation, consider the invariant case with $k=1$. In this case, the linear invariant mapping $g : \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}$ is essentially an invariant pooling (Def.5.7). As $b(1)=1$, the linear invariant mapping $g : \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}$ only holds one single base — the sum pooling, *i.e.,* g follows the form $g(a) = c\langle \mathbf{1}, a\rangle$, where c is a parameter to be learned. Consider the equivariant case, where $k=1$ and $k'=1$. As $b(2)=2$, the linear equivariant mapping $g : \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}^{|\mathcal{V}|}$ holds two bases, *i.e.,* g has the form $g(a) = (c_1 I + c_2 \mathbf{1}\mathbf{1}^\top)a$, where c_1, c_2 are parameters to be learned.

Based on the above observations, GNNs can be built by compositing these linear invariant/equivariant mappings. Learning can be performed via learning the weights before the above bases. Towards this end, Maron et al (2018, 2019a) has proposed using these linear invariant/equivariant mappings to build GNNs:

$$f_{k-inv} = g_{inv} \circ g_{equ}^{(L)} \circ \sigma \circ g_{equ}^{(L-1)} \circ \sigma \cdots \circ \sigma \circ g_{equ}^{(1)}, \tag{5.21}$$

where g_{inv} is a linear invariant layer $\mathbb{R}^{\otimes_k|\mathcal{V}|} \to \mathbb{R}$, $g_{equ}^{(l)}$'s, $l \in [L]$ are linear equivariant layers from $\mathbb{R}^{\otimes_k|\mathcal{V}|} \to \mathbb{R}^{\otimes_k|\mathcal{V}|}$, and σ is an element-wise non-linear activation function. It can be shown that f_{k-inv} is an invariant mapping. Maron et al (2018); Azizian and Lelarge (2020) proved that the connection of f_{k-inv} to the k-WL test can be summarized with the following theorem.

Theorem 5.12. *(Reproduced from (Maron et al, 2018; Azizian and Lelarge, 2020)) For two non-isomorphic graphs $\mathcal{G}^{(1)} \ncong \mathcal{G}^{(2)}$, if the k-WL test can distinguish them, then there exists f_{k-inv} that can distinguish them.*

Maron et al (2019b); Keriven and Peyré (2019) also studied whether the models f_{k-inv} may universally approximate any permutation invariant function. However, they were pessimistic in their conclusion since this would require high-order tensors, $k = \Omega(n)$, which can hardly be implemented in practice (Maron et al, 2019b).

Similar to k-GNN, f_{inv} is also at most as powerful as the k-WL test. To be more expressive than MP-GNN, f_{inv} should use at least $k=3$. Therefore, the memory complexity is at least $\Omega(|\mathcal{V}|^3)$. Then, the number of bases of the linear equivariant layer is $b(6) = 203$. Therefore, the computation at each layer follows that: (1) a tensor in $\mathbb{R}^{\otimes_3|\mathcal{V}|}$ times $b(6)$ many tensors in $\mathbb{R}^{\otimes_6|\mathcal{V}|}$ get $b(6)$ many tensors in $\mathbb{R}^{\otimes_3|\mathcal{V}|}$; (2) these tensors get summed into a single tensor in $\mathbb{R}^{\otimes_3|\mathcal{V}|}$.

5.4.4.3 k-FWL-induced GNNs (Maron et al, 2019a; Chen et al, 2019f)

The higher-order GNNs in previous two subsections match the expressive power of the k-WL test. According to Theorem 5.11, the k-FWL test holds the same power as the $k+1$-WL test, which is strictly more powerful than the k-WL test for $k \geq 2$, while the k-FWL test only needs to track the representations of k-tuples. Therefore,

if GNNs can mimic the k-FWL test, they may hold similar memory cost as the GNNs introduced in the previous two subsections while being more expressive. Maron et al (2019a); Chen et al (2019f) proposed PPGN and Ring-GNN respectively to match the k-FWL test.

The key difference between the k-FWL test and the k-WL test is the leveraging of the neighbors of a k-tuple V_j. Note that $N_{k-FWL}(V_j;u)$ in Eq.equation 5.18 groups the neighboring tuples of V_j into a higher-level tuple, while $N_{k-WL}(V_j;u)$ skips grouping them due to the set union operation in Eq.equation 5.19. This yields the key mechanism in the GNN design to match the k-FWL test: implement the aggregating procedure in the k-FWL test of Eq.equation 5.18 via a product-sum procedure. Suppose the representation for V_j is $\mathbf{h}_j \in \mathbb{R}$. We may design the aggregation of $\{(C^{(l)}_{j'}|V_{j'} \in N_{k-FWL}(V_j;u))\}_{u\in V}$ as

$$\sum_{u\in V} \prod_{V_{j'}\in N_{k-FWL}(V_j;u)} \mathbf{h}_{j'}.$$

If we combine all these representations into a tensor $H \in \mathbb{R}^{\otimes_k |V|\times F}$, the above operation can essentially be represented as tensor operation, i.e., define

$$H' := \sum_{u\in V} H_{u,\cdot,\cdots,\cdot} \odot H_{\cdot,u,\cdots,\cdot} \odot \cdots \odot H_{\cdot,\cdot,\cdots,u}, \text{ where}$$
$$[H']_{v_{j_1},v_{j_2},\cdots,v_{j_k}} = \sum_{u\in V} H_{u,v_{j_2},\cdots,v_{j_k}} \cdot H_{v_{j_1},u,\cdots,v_{j_k}} \cdot \cdots \cdot H_{v_{j_1},v_{j_2},\cdots,u}.$$

Based on the above observation, Maron et al (2019a) built PPGN as follows. First, for all $V_j \in \mathscr{V}^k$, initialize $\mathbf{h}^{(0)}_j \in \mathbb{R}$ such that $\mathbf{h}^{(0)}_j = \mathbf{h}^{(0)}_{j'}$, if and only if: (1) $X_{v_{ja}} = X_{v_{j'_a}}$; (2) $v_{ja} = v_{jb} \Leftrightarrow v_{j'_a} = v_{j'_b}$; and (3) $(v_{ja}, v_{j'_b}) \in \mathscr{E} \Leftrightarrow (v_{j'_a}, v_{j'_b}) \in \mathscr{E}$, for all $a,b \in [k]$. Then, combine $\mathbf{h}^{(0)}_j$ into a tensor $H^{(0)} \in \mathbb{R}^{\otimes_k |\mathscr{V}|}$. Perform the updating procedure for $l = 0,1,...,L-1$:

$$H^{(l+1)} = \tilde{H}^{(l,0)} \oplus \left[\sum_{u\in V} \tilde{H}^{(l,1)}_{u,\cdot,\cdots,\cdot} \odot \tilde{H}^{(l,2)}_{\cdot,u,\cdots,\cdot} \odot \cdots \odot \tilde{H}^{(l,k)}_{\cdot,\cdot,\cdots,u}\right],$$
$$\text{where,} \quad \tilde{H}^{(l,i)} = \text{MLP}^{(l,i)}(H^{(l)}). \tag{5.22}$$

Here, MLPs are imposed at the last dimension of these tensors. MLPs with different sup-script have different parameters. Finally, perform a READOUT $\sum_{V_j\in V^k} \mathbf{h}^{(L)}_j$ to obtain the graph representation.

Maron et al (2019a) proved that PPGN, when $k = 2$, can match the power of the 2-FWL test. Azizian and Lelarge (2020) generalized this result to an arbitrary k.

Theorem 5.13. *(Reproduced from (Azizian and Lelarge, 2020)) For two non-isomorphic graphs $G^{(1)} \ncong G^{(2)}$, if the k-FWL test can distinguish them, then there exists a PPGN that can distinguish them.*

To be more powerful than the 1-WL test, PPGN only needs to set $k = 2$ and hence the memory complexity is just $\Omega(|\mathcal{V}|^2)$. Regarding the computation, the product-sum-type aggregation of PPGN is indeed more complex than f_{inv} in Sec.5.4.4.2. However, when $k = 2$, Eq.equation 5.22 reduces to the product of two matrices, which can be efficiently computed in parallel computing units.

5.5 Summary

Graph neural networks have recently achieved unprecedented success across many domains due to their great expressive power to learn complex functions defined over graphs and relational data. In this chapter, we provided a systematic study of the expressive power of GNNs by giving an overview of recent research results in this field.

We first established that the message passing GNN is at most as powerful as the 1-WL test to distinguish non-isomorphic graphs. The key condition that guarantees to match the limit is an injective updating function of node representations. Next, we discussed techniques that have been proposed to build more powerful GNNs. One approach to make message passing GNNs more expressive is to pair the input graphs with extra attributes. In particular, we discussed two types of extra attributes — random attributes and deterministic distance attributes. Injecting random attributes allows GNNs to distinguish any non-isomorphic graphs, though a large amount of data augmentation is required to make GNNs approximately invariant. Meanwhile, injecting deterministic distance attributes does not require the same data augmentation, but the expressive power of the resulting GNNs still holds certain limitations. Mimicking higher-dimensional WL tests is another way to build more powerful GNNs. These approaches do not track node representations. Instead, they update the representation of every k-tuple of nodes ($k \geq 2$). Overall, the message passing GNN is powerful but holds some limits in its expressive power. Different techniques make GNNs overcome these limits to a different extent while incurring different types of computational costs.

We would like to list some additional research results on the expressive power of GNNs that we were not able to cover earlier due to space limitations. Barceló et al (2019) study the expressive power of GNNs to represent Boolean classifiers, which is useful to understand how GNNs represent knowledge and logic. Vignac et al (2020a) propose a structural message passing framework for GNNs, where a matrix instead of a vector is adopted as the node representation to make GNN more expressive. Balcilar et al (2021) studied the expressivity of GNNs via the spectral analysis of GNN-based graph signal transformations. Chen et al (2020k) studies the effect of non-linearity of GNNs in the message passing procedure on their expressive power, which complements our understanding of many works that suggest a linear message passing procedure (Wu et al, 2019a; Klicpera et al, 2019a; Chien et al, 2021).

Theoretical characterization of GNNs is an important research direction, where the analysis of expressive power is only one of its aspects, perhaps the best-studied up to this point. Machine learning models hold two fundamental blocks, training and generalization. However, only a few research works have analyzed them (Garg et al, 2020; Liao et al, 2021; Xu et al, 2020c). The authors suggest that future research on building more expressive GNNs always takes these two blocks into account. A related, significant question is how to build more expressive GNNs with only a limited depth and width[4]. Note that limiting the model depth and width yields the potential of more efficient GNN training and better generalization. To conclude this chapter, let us quote Sir Winston Churchill:"Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning." We have strong confidence that the machine learning community will put more effort on theory for GNNs in the future to match their success and break their encountered difficulties in real-world applications.

Acknowledgements The authors would like to greatly thank Jiaxuan You and Weihua Hu for sharing many materials reproduced here. The authors would like to greatly thank Rok Sosič and Natasha Sharp for commenting on and polishing the manuscript. The authors also gratefully acknowledge the support of DARPA under Nos. HR00112190039 (TAMI), N660011924033 (MCS); ARO under Nos. W911NF-16-1-0342 (MURI), W911NF-16-1-0171 (DURIP); NSF under Nos. OAC-1835598 (CINES), OAC-1934578 (HDR), CCF-1918940 (Expeditions), IIS-2030477 (RAPID), NIH under No. R56LM013365; Stanford Data Science Initiative, Wu Tsai Neurosciences Institute, Chan Zuckerberg Biohub, Amazon, JPMorgan Chase, Docomo, Hitachi, Intel, JD.com, KDDI, NVIDIA, Dell, Toshiba, Visa, and UnitedHealth Group. J. L. is a Chan Zuckerberg Biohub investigator.

Editor's Notes: The theoretical analysis of expressive power reveals how the architecture of GNNs works and gains its advantage. Hence it provides support for readers to understand the great success of GNNs in fundamental graph learning tasks, e.g. link prediction (chapter 10) and graph matching (chapter 13), various downstream tasks, e.g. recommender system (chapter 19) and natural language processing (chapter 21), as well as its relevance with other GNNs' characterizations, e.g. scalability (chapter 6) and robustness (chapter 8). Inspired by these theories, it's also probably to motivate the study of preferable GNN models that can break through unsolved challenges in existing problems, such as graph transformation (chapter 12) and drug discovery (chapter 24).

[4] Loukas (2020) measures the required depth and width of GNNs by viewing them as distributed algorithms, which does not assume permutation invariance. Instead, here we are talking about the expressive power that refers to the capability of learning permutation invariant functions.