Survey on Generalization Theory for Graph Neural Networks

Antonis Vasileiou[1], Stefanie Jegelka[2], Ron Levie[3], and Christopher Morris[1]

[1]RWTH Aachen University
[2]Technical University of Munich & Massachusetts Institute of Technology
[3]Technion - Israel Institute of Technology

Message-passing graph neural networks (MPNNs) have emerged as the leading approach for machine learning on graphs, attracting significant attention in recent years. While a large set of works explored the expressivity of MPNNs, i.e., their ability to separate graphs and approximate functions over them, comparatively less attention has been directed toward investigating their generalization abilities, i.e., making meaningful predictions beyond the training data. Here, we systematically review the existing literature on the generalization abilities of MPNNs. We analyze the strengths and limitations of various studies in these domains, providing insights into their methodologies and findings. Furthermore, we identify potential avenues for future research, aiming to deepen our understanding of the generalization abilities of MPNNs.

Contents

1. Introduction

Graphs model interactions among entities across the life, natural, and formal sciences, such as atomistic systems [Duval et al., 2023, Zhang et al., 2023] or social networks [Easley and Kleinberg, 2010, Lovász, 2012], underlining the need for machine learning methods to handle graph-structured data. Hence, neural networks designed for graph-structured data, mainly *message-passing graph neural networks* (MPNNs) [Merkwirth and Lengauer, 2005, Gori et al., 2005, Hamilton et al., 2017, Kipf and Welling, 2017, Gilmer et al., 2017, Scarselli et al., 2009], have gained significant attention in the machine learning community, showcasing promising outcomes across diverse domains [Corso et al., 2024], spanning drug design [Wong et al., 2023], global medium-range weather forecasting [Lam et al., 2023], and combinatorial optimization [Cappart et al., 2021, Gasse et al., 2019, Qian et al., 2023].[1]

While MPNNs are successful in practice and make a real-world impact, their theoretical properties are understood to a lesser extent [Morris et al., 2024]. Only MPNNs' *expressivity* is somewhat understood. Here, the expressivity of MPNNs is modeled mathematically based on two main approaches, algorithmic alignment with the 1-*dimensional Weisfeiler–Leman algorithm* (1-WL) and universal approximation theorems [Azizian and Lelarge, 2021, Böker et al., 2023, Chen et al., 2019, Geerts and Reutter, 2022, Maehara and NT, 2019, Rauchwerger et al., 2024]. Here, the 1-WL [Weisfeiler and Leman, 1968, Weisfeiler, 1976, Morris et al., 2021] is a well-studied heuristic for the graph isomorphism problem. Concretely, regarding the former, Morris et al. [2019a], Xu et al. [2019a] showed that the 1-WL limits the expressivity of any possible MPNN architecture in distinguishing non-isomorphic graphs.

---

[1]We use the term MPNNs to refer to graph-machine learning architectures that fit into the framework of Gilmer et al. [2017], see Section 2.1, and use the term *graph neural networks* (GNNs) in a broader sense, i.e., all neural network architectures capable of handling graph-structured inputs.

In contrast, less is known about MPNNs' abilities to make meaningful predictions beyond the training data. More precisely, the *generalization* abilities of MPNNs assess the architectures' ability to adapt effectively to new, previously unseen graphs originating from the same distribution as the training set. In addition, *extrapolation* or *out-of-distribution generalization* distinguishes itself by involving unseen graphs drawn from a (slightly) different distribution than the training set. Nowadays, there exist several analyses of MPNNs' generalization using different theoretical frameworks such as *Vapnik–Chervonenkis dimension* (VC dimension) [Morris et al., 2023, Scarselli et al., 2018], *Rademacher averages* [Garg et al., 2020], and related formalism. However, the resulting bounds are often hard to compare due to different assumptions, MPNN architectures, and parameters.

1.1. Contributions

Here, we survey theoretical results on MPNNs' generalization abilities, making it easier to compare different results and quickly get into the field. We survey generalization results based on VC dimension, Rademacher complexity, covering number bounds, stability-based generalization bounds, and PAC-Bayesian analysis. In addition, we also look into generalization analysis using graphon theory and generalization theory for node-level prediction tasks and out-of-distribution generalization. Our discussion covers the mathematical tools used to establish these bounds and the theoretical foundations upon which they rely. The main goal of this survey is to provide readers with a comprehensive overview of MPNNs' generalization abilities and offer insights into potential paths for extending the current theory to address gaps in the literature. In addition, we propose open problems and future challenges in the field to help spur future work.

1.2. Related work

As mentioned above, there exists a large set of work relating MPNNs' expressive power to the 1-WL [Azizian and Lelarge, 2021, Böker et al., 2023, Geerts and Reutter, 2022, Maron et al., 2019, Morris et al., 2019a, Xu et al., 2019a] and its generalizations; see Morris et al. [2021] for a thorough overview. Regarding MPNNs' expressiveness and generalization, Jegelka [2022] provides a good overview of works until 2022. In addition, in their position paper, Morris et al. [2024] argue that the current emphasis on 1-WL-based expressivity analysis is limited and outline critical challenges regarding a fine-grained analysis of expressive power, generalization, and optimization dynamics of MPNNs and related graph learning architectures.

2. Background

Let $\mathbb{N} \coloneqq \{1, 2, \dots\}$ and $\mathbb{N}_0 \coloneqq \mathbb{N} \cup \{0\}$. The set $\mathbb{R}^+$ denotes the set of non-negative real numbers. For $n \in \mathbb{N}$, let $[n] \coloneqq \{1, \dots, n\} \subset \mathbb{N}$. We use $\{\!\{\dots\}\!\}$ to denote multisets, i.e., the generalization of sets allowing for multiple, finitely many instances for each of its elements. For two non-empty sets X and Y, let Y^X denote the set of functions from X to Y. Let $S \subset \mathbb{R}^d$, then the *convex hull* $\mathsf{conv}(S)$ is the minimal convex set containing the set S. Let $\boldsymbol{M}$ be an $n \times m$ matrix, $n > 0$ and $m > 0$, over $\mathbb{R}$, then $\boldsymbol{M}_{i,\cdot}$, $\boldsymbol{M}_{\cdot,j}$, $i \in [n]$, $j \in [m]$, are the ith row and jth column, respectively, of the matrix $\boldsymbol{M}$. For $\boldsymbol{x} \in \mathbb{R}^{1 \times d}$, we use $\|\cdot\|_1$ and $\|\cdot\|_2$ to refer to the 1-*norm* $\|\boldsymbol{x}\|_1 \coloneqq |x_1| + \cdots + |x_d|$ and 2-*norm* $\|\boldsymbol{x}\|_2 \coloneqq \sqrt{x_1^2 + \cdots + x_d^2}$, respectively. For $\boldsymbol{p} \in \mathbb{R}^{1 \times d}, d > 0$, and $\varepsilon > 0$, the *ball* $B_{\|\cdot\|}(\boldsymbol{p}, \varepsilon, d) \coloneqq \{\boldsymbol{s} \in \mathbb{R}^{1 \times d} \mid \|\boldsymbol{p} - \boldsymbol{s}\| \leq \varepsilon\}$; when the norm is evident from

Table 1: Summary of results investigating MPNNs' generalization abilities, categorized by graph-level or node-level tasks, the different frameworks employed, and the types of MPNN architectures, i.e., general MPNNs or graph convolutional neural networks.

	Framework	Architectures	References
Node-level	Rademacher complexity	GCNs	[Lv, 2021]
	Stability-based	GCNs	[Verma and Zhang, 2019, Zhou and Wang, 2021]
	VC dimension	MPNNs	[Scarselli et al., 2018]
	Transductive learning	MPNNs	[Cong et al., 2021, Esser et al., 2021, Oono and Suzuki, 2020, Tang and Liu, 2023]
Graph-level	Rademacher complexity	GCNs	[Lv, 2021]
	PAC-Bayesian	GCNs	[Ju et al., 2023, Liao et al., 2021, Sun and Lin, 2024, Wu et al., 2023b]
	VC dimension	MPNNs	[D'Inverno et al., 2024, Franks et al., 2023, Morris et al., 2023, Scarselli et al., 2018]
	Rademacher complexity	MPNNs	[Garg et al., 2020, Karczewski et al., 2024]
	PAC-Bayesian	MPNNs	[Ju et al., 2023, Liao et al., 2021, Sun and Lin, 2024]
	Covering number	MPNNs	[Levie, 2023, Maskey et al., 2022, 2024, Rauchwerger et al., 2024, Vasileiou et al., 2024]
	Out-of-distribution	MPNNs	[Bevilacqua et al., 2021, Gui et al., 2023, Li et al., 2022b, 2024b, Xu et al., 2021, Yehudai et al., 2021]

the context, we omit the subscript. In what follows, **o** denotes an all-zero vector with an appropriate number of components.

Graphs An *(undirected) graph* G is a pair $(V(G), E(G))$ with *finite* sets of *vertices* or *nodes* $V(G)$ and *edges* $E(G) \subseteq \{\{u, v\} \subseteq V(G) \mid u \neq v\}$. For ease of notation, we denote an edge $\{u, v\}$ in $E(G)$ by (u, v) or (v, u). The *order* of a graph G is its number $|V(G)|$ of vertices. If not stated otherwise, we set $n := |V(G)|$ and call G an *n-order graph*. We denote the set of all n-order graphs by $\mathcal{G}$. For an n-order graph G, we denote its *adjacency matrix* by $\boldsymbol{A}(G) \in \{0,1\}^{n \times n}$, where $\boldsymbol{A}(G)_{vw} = 1$ if and only if $(v, w) \in E(G)$. We further define the Laplacian matrix $\boldsymbol{L}(G) \in \mathbb{R}^{n \times n}$ as $\boldsymbol{L}(G) := \boldsymbol{D}(G) - \boldsymbol{A}(G)$, where $\boldsymbol{D}(G) \in \mathbb{R}^{n \times n}$ is a degree diagonal matrix with diagonal element $D_{ii} = \sum_j a_{ij}$, for $i \in [n]$. The *neighborhood* of $v \in V(G)$ is denoted by $N(v) := \{u \in V(G) \mid (v, u) \in E(G)\}$ and the *degree* of a vertex v is $|N(v)|$. A *(vertex-)labeled graph* is a pair (G, ℓ_G) with a graph $G = (V(G), E(G))$ and a (vertex-)label function $\ell_G \colon V(G) \to \Sigma$, where Σ is an arbitrary label set. For a vertex $v \in V(G)$, $\ell_G(v)$ denotes its *label*. A Boolean *(vertex-)d-labeled graph* is a pair (G, ℓ_G) with a graph $G = (V(G), E(G))$ and a label function $\ell_G \colon V(G) \to \{0,1\}^d$. We denote the set of all n-order Boolean d-attributed graphs by $\mathcal{G}_{n,d}^{\mathbb{B}}$. An *attributed graph* is a pair (G, a_G) with a graph $G = (V(G), E(G))$ and an (vertex-)attribute function $a_G \colon V(G) \to \mathbb{R}^{1 \times d}$, for $d > 0$. The attribute or feature of $v \in V(G)$ is $a_G(v)$. We denote the class of all n-order graphs with d-dimensional, real-valued vertex features by $\mathcal{G}_{n,d}^{\mathbb{R}}$. In addition, we denote the class of all graphs up to order n with d-dimensional, real-valued vertex features by $\mathcal{G}_{n,d}^{\mathbb{R}}$. The class of all graphs is denoted by $\mathcal{G}$.

Two graphs G and H are *isomorphic* if there exists a bijection $\varphi \colon V(G) \to V(H)$ that preserves adjacency, i.e., $(u, v) \in E(G)$ if and only if $(\varphi(u), \varphi(v)) \in E(H)$. In the case of attributed graphs, we additionally require $\ell_G(v) = \ell_H(\varphi(v))$ for all $v \in V(G)$. Given two graphs G and H with disjoint vertex sets, we denote their disjoint union by $G \,\dot\cup\, H$.

2.1. Message-passing graph neural networks

Message-passing graph neural networks (MPNNs) are a neural network framework for graph inputs. Intuitively, MPNNs learn a vectorial representation, i.e., d-dimensional real-valued vector, representing each vertex in a graph by aggregating information from neighboring vertices. Formally, let (G, ℓ_G) be a labeled graph with initial vertex features $\boldsymbol{h}_v^{(0)} \in \mathbb{R}^d$ that are *consistent* with ℓ. That is, each vertex v is annotated with a feature $\boldsymbol{h}_v^{(0)} \in \mathbb{R}^d$ such that $\boldsymbol{h}_v^{(0)} = \boldsymbol{h}_u^{(0)}$ if and only if $\ell_G(v) = \ell_G(u)$. An example is a one-hot encoding of the labels $\ell(u)$ and $\ell(v)$. Alternatively, $\boldsymbol{h}_v^{(0)}$ can be an attribute or a feature of the vertex v, e.g., physical measurements of atoms of chemical molecules. An MPNN architecture consists of a stack of neural network layers, i.e., a composition of permutation-equivariant parameterized functions. Each layer aggregates local neighborhood information, i.e., the neighbors' features around each vertex, and then passes this aggregated information on to the next layer. Following Gilmer et al. [2017] and Scarselli et al. [2009], in each layer $t > 0$, we compute vertex features

$$\boldsymbol{h}_v^{(t)} := \mathsf{UPD}^{(t)}\Big(\boldsymbol{h}_v^{(t-1)}, \mathsf{AGG}^{(t)}\big(\{\!\!\{(\boldsymbol{h}_v^{(t-1)}, \boldsymbol{h}_u^{(t-1)}) \mid u \in N(v)\}\!\!\}\big)\Big) \in \mathbb{R}^{d_t}, \tag{1}$$

for $v \in V(G)$, where $\mathsf{UPD}^{(t)}$ and $\mathsf{AGG}^{(t)}$ may be parameterized functions, e.g., neural networks. In the case of graph-level tasks, e.g., graph classification, one uses a *readout layer*

$$\boldsymbol{h}_G := \mathsf{READOUT}\big(\{\!\!\{\boldsymbol{h}_v^{(L)} \mid v \in V(G)\}\!\!\}\big) \in \mathbb{R}^d, \tag{2}$$

to compute a single vectorial representation based on learned vertex features after iteration L.

Finally, the output of the readout layer is passed through a parameterized function, typically a feed-forward neural network $\mathsf{FNN}\colon \mathbb{R}^d \to \mathbb{R}$, which performs the final classification or regression. The $\mathsf{READOUT}$ layer is not used for node-level tasks, and we apply the parameterized function FNN directly to the vertex features for predicting the node labels. The parameters of UPD, AGG, $\mathsf{READOUT}$, and FNN are optimized end-to-end, usually through a variant of *(stochastic) gradient-descent* (SGD), e.g., [Kingma and Ba, 2015], against a loss function.

2.2. Example of MPNN layers

Here, we define various MPNN layers used in this survey.

Simple MPNNs For a given d and $L \in \mathbb{N}$, we define the class $\mathsf{MPNN}_{\mathsf{FNN}}(d, L)$ of simple MPNNs as L-layer MPNNs for which, in Equation (1), for each $t \in [L]$, the aggregation function $\mathsf{AGG}^{(t)}$ is summation and the update function $\mathsf{UPD}^{(t)}$ is a feed-forward neural network $\mathsf{FNN}^{(t)}\colon \mathbb{R}^{2d} \to \mathbb{R}^d$ of width at most d. Similarly, the readout function in Equation (2) consists of a feed-forward neural network $\mathsf{FNN}\colon \mathbb{R}^d \to \mathbb{R}^d$ applied on the sum of all vertex features computed in layer L.[2] More specifically, MPNNs in $\mathsf{MPNN}_{\mathsf{FNN}}(d, L)$ compute on an attributed graph (G, ℓ_G) with d-dimensional initial vertex features $\boldsymbol{h}_v^{(0)} \coloneqq a_G(v) \in \mathbb{R}^d$, for $v \in V(G)$, the following vertex features, for each $v \in V(G)$,

$$\boldsymbol{h}_v^{(t)} \coloneqq \mathsf{FNN}^{(t)}\Big(\boldsymbol{h}_v^{(t-1)}, \sum_{u \in N(v)} \boldsymbol{h}_u^{(t-1)}\Big) \in \mathbb{R}^d,$$

for $t \in [L]$, and

$$\boldsymbol{h}_G \coloneqq \mathsf{FNN}\Big(\sum_{v \in V(G)} \boldsymbol{h}_v^{(L)}\Big) \in \mathbb{R}^d.$$

The class $\mathsf{MPNN}_{\mathsf{FNN}}(d, L)$ encompasses the GNN architecture derived by Morris et al. [2019a] that has the same expressive power as the 1-WL in distinguishing non-isomorphic graphs. In addition, by $\mathsf{MPNN}(L)$, we denote the set of all L-layer simple MPNNs.

Graph convolutional networks *Graph convolutional networks* (GCNs) are a special case of MPNNs [Kipf and Welling, 2017]. We consider an undirected graph $G \in \mathcal{G}_{n,d}^{\mathbb{R}}$ with adjacency matrix $\boldsymbol{A}$, graph Laplacian $\boldsymbol{L}$, and node features $\mathbf{h}_v \in \mathbb{R}^d$, for $v \in V(G)$. We further assume that $V(G) \coloneqq [n]$, and we also add self-connections for each node, i.e., $A_{ii} = 1$, for $i \in [n]$. We define a graph filter $g(\boldsymbol{L}) \in \mathbb{R}^{n \times n}$ as a function of the graph Laplacian $\boldsymbol{L}$, where typically $g(\boldsymbol{L}) \coloneqq \boldsymbol{A} + \mathbf{I}_n$ with $\mathbf{I}_n$ being the $n \times n$ identity matrix, or a Chebyshev polynomial of $\boldsymbol{L}$. Since the subsequent results focus solely on GCNs with a single layer and GCNs with a single hidden layer followed by an output layer with a single neuron, we exclusively present these two architectures. For the general architecture of k-layer GCNs, please refer to the Appendix C. In the single-layer architecture, for a vertex v, we compute

$$\boldsymbol{h}_v \coloneqq \sigma\left(\sum_{u \in N(v)} [g(\boldsymbol{L})]_{\cdot u} \boldsymbol{h}_u^{\intercal} \boldsymbol{\theta}\right)$$

---

[2]For simplicity, we assume that all feature dimensions of the layers are fixed to $d \in \mathbb{N}$.

where σ represents a component-wise non-linear activation function, $[g(\boldsymbol{L})]._u$ denotes the uth column of the graph filter, and $\boldsymbol{\theta} \in \mathbb{R}$ is a learnable parameter. Considering the single hidden layer with an output layer with a single neuron architecture for a vertex v,

$$\boldsymbol{h}_v := \sigma\left(\sum_{t=1}^{k} w_t^{(2)} \sum_{u=1}^{n} [g(\boldsymbol{L})]_{vu} \cdot \sigma\left(\sum_{l=1}^{d} w_{lt}^{(1)} \sum_{j=1}^{n} [g(\boldsymbol{L})]_{uj} (\boldsymbol{h}_u)_l\right)\right), \tag{3}$$

where $w_t^{(2)}$ and $w_{i,j}^{(1)}$ are learnable parameters and $(\mathbf{x}_u)_l$ is the l-th entry of the vector $\mathbf{x}_u$.

2.3. Supervised learning on graphs and generalization

In the following, we introduce the formal framework of supervised learning on graphs.

2.3.1. The statistical learning setting

In *supervised learning* on graphs, we are given a set of graphs, e.g., the set of graphs $\mathcal{G}_d^{\mathbb{R}}$ with d-dimensional, real-valued vertex features, and a set of *labels* $\mathcal{Y}$. For example, in *binary classification*, the label set is typically represented by the set $\{0, 1\}$, signifying that each graph is categorized into one of two classes. In the case of *regression*, the label set is $\mathcal{Y} := \mathbb{R}$. The primary goal of using MPNNs for learning on graphs is to estimate a function $f : \mathcal{G}_d^{\mathbb{R}} \to \mathcal{Y}$, where we represent f through an MPNN architecture and a feed-forward neural network as described in Section 2, i.e., $f(G) := \mathsf{FNN}(\boldsymbol{h}_G)$. Such a function f is commonly referred to as an *MPNN concept* or *concept*. We denote $\mathcal{H} := \{f : \mathcal{G}_d^{\mathbb{R}} \to \mathcal{Y} \mid f \text{ is an MPNN concept}\}$ as the set of all possible MPNN concepts or the *hypothesis class*.

To find a suitable concept f, we access a finite set of *training points* $\mathcal{S} := \{(X, y) \mid (X, y) \in \mathcal{G}_d^{\mathbb{R}} \times \mathcal{Y}\}$. A *learning algorithm* is a procedure that takes the training data as input and outputs a concept f. Additionally, we assume a joint probability distribution P on $\mathcal{G}_d^{\mathbb{R}} \times \mathcal{Y}$, with training examples $(X, y) \in \mathcal{S}$ sampled independently from the distribution P. This type of sampling is often denoted as *independent and identically distributed* (i.i.d.) sampling. Furthermore, to appropriately estimate an MPNN concept, we need a function measuring how close the predicted label is to the true label. To formally measure the goodness of our concept, we introduce a loss function $\ell : \mathcal{H} \times \mathcal{G}_d^{\mathbb{R}} \times \mathcal{Y} \to \mathbb{R}^+$. For example, in the case of regression, the *squared error*

$$\ell(f, G, y) := |f(G) - y|^2$$

is commonly used. Now, for an MPNN concept f, and a loss function ℓ we define the *expected loss* as

$$\ell_{\exp}(f) := \mathbb{E}_{(G,y) \sim P}[\ell(f, G, y)].$$

While the loss function measures the error of a concept on individual data points, the *expected loss* is the average loss over data points generated according to distribution P. Therefore, a concept is better than another if it has a smaller *expected loss*. Hence, the objective is to minimize the *expected loss* to find a concept capable of accurately predicting labels of samples from the distribution P. However, since the distribution P of the data is unknown to the learning algorithm, we rely on the *empirical loss* estimating the *expected loss*, i.e., the average loss over the training data,

$$\ell_{\mathrm{emp}}(f) := \frac{1}{N} \sum_{i=1}^{n} \ell(f, X_i, y_i).$$

Our goal is to find a concept $f_{\mathcal{S}}$ minimizing the empirical loss. Typically, the *empirical loss* $\ell_{\mathrm{emp}}(f)$ for a function f learned from a specific training set is relatively low, indicating its ability to capture the training data. Yet, it remains uncertain whether a function f with small error on the training set also performs well across the rest of the input space $\mathcal{G}$, as reflected in its expected loss $\ell_{\mathrm{exp}}(f)$. Intuitively, an MPNN concept f generalizes effectively when the difference $|\ell_{\mathrm{exp}}(f) - \ell_{\mathrm{emp}}(f)|$ is small. This difference is the *generalization gap* of the concept f.

Node-level prediction Defining the generalization gap for node-level prediction tasks bears some issues. In this setting, as described in Section 2.1, we do not use a readout layer, and instead, we directly apply a feed-forward neural network to a node's output feature vector to predict the node's label from $\mathcal{Y}$. Therefore, defining the empirical loss using a loss function ℓ for node-level tasks, e.g., classification or regression, is straightforward. However, for the expected loss, it is necessary to define the space on which we assume a distribution under which the expectation in Section 2.3.1 is computed. Different approaches have been proposed in the literature. For example, Scarselli et al. [2018] described a framework where the underlying distribution P is used for i.i.d. sampling triplets (G, u, y) consisting of a graph $G \in \mathcal{G}_d$, a node $u \in V(G)$, and a label $y \in \mathcal{Y}$. Alternatively, in the framework developed by Verma and Zhang [2019], considering a fixed graph $G \in \mathcal{G}_d^{\mathbb{R}}$, the underlying distribution P is used for i.i.d. sampling a pair (v, y) consisting of a node $v \in G$ and a label $y \in \mathcal{Y}$. It is worth noting that in the node classification setting, while nodes are often assumed to be sampled in an i.i.d. fashion from $V(G)$, this assumption is neither widely accepted nor particularly intuitive. Consequently, several works have avoided this assumption, instead adopting a transductive learning approach to define the generalization gap, as described in Section 8.

Generalization and concentration inequalities At first glance, the difference we aim to bound, i.e., $|\ell_{\mathrm{emp}}(f) - \ell_{\mathrm{exp}}(f)|$, appears to be the deviation of a random variable $\ell_{\mathrm{emp}}(f)$ from its expectation $ell_{\mathrm{exp}}(f)$. Since the observations are i.i.d., one might initially conjecture that a standard *concentration inequality* (e.g., Hoeffding's inequality, see [Hoeffding, 1963]) could be applied to bound this difference. Specifically, for a fixed function f with an appropriate domain and under mild assumptions, a simple variant of Hoeffding's inequality provides an upper bound (with high probability) on the deviation $\left| \sum_{i=1}^{n} f(X_i) - \mathbb{E}[f(X_1)] \right|$, where X_i are i.i.d. random variables. However, if we look more carefully, we observe that in $\ell_{\mathrm{emp}}(f)$, the terms in the summation are not independent. This comes from the fact that the function f is selected based on the training sample, thereby introducing dependence between f and $\mathcal{S}$. Consequently, Hoeffding's inequality is not directly applicable. The main two approaches in generalization theory either employ concentration inequalities to derive bounds that hold uniformly over all functions f in a hypothesis class, e.g., VC bounds (see Section 3) or Rademacher complexity bounds (Section 4), or endow the hypothesis class with probability distributions and derive bounds for the expected difference concerning the induced distribution (PAC-Bayesian approach, see Section 6).

3. MPNN generalization bounds via VC dimension

The VC dimension, introduced in Vapnik and Chervonenkis [1964], is one example for capturing the capacity of the hypothesis class $\mathcal{H}$ for binary classification problems. The key contribution of VC dimension theory [Vapnik and Chervonenkis, 1964, Vapnik, 1995, 1998] lies in bounding

the generalization gap by the worst-case behavior across all MPNN concepts f in a hypothesis class $\mathcal{H}$ that the learning algorithm could select, noting that the actual trained network must be one of these MPNNs. In other words, instead of bounding the difference $|\ell_{\text{emp}}(f) - \ell_{\text{exp}}(f)|$ for the trained classifier $f = f_{\mathcal{S}}$, the VC dimension aims to bound the supremum

$$\sup_{f \in \mathcal{H}} |\ell_{\text{emp}}(f) - \ell_{\text{exp}}(f)|.$$

In the following, we introduce the VC dimension tailored to graphs and MPNNs. Let us consider a binary classification problem over the graph class $\mathcal{G}_d^{\mathbb{R}}$, for $d \in \mathbb{N}$, and a hypothesis class $\mathcal{H}$ of MPNN concepts from $\mathcal{G}_d^{\mathbb{R}}$ to $\{0, 1\}$. We say that the graphs $\{G_1, \ldots, G_N\} \subseteq \mathcal{G}_d^{\mathbb{R}}$ are shattered by the hypothesis class $\mathcal{H}$ if for any $\boldsymbol{\tau} \in \{0, 1\}^N$ there exists a classifier $f \in \mathcal{H}$ such that for all $i \in [N]$,

$$f(X_i) = \begin{cases} 0, & \text{if } \tau_i = 1, \text{ and} \\ 1, & \text{if } \tau_i = 1. \end{cases}$$

For a set $\mathcal{X}$ of graphs and a class of MPNN concepts $\mathcal{H}$, the VC dimension, denoted by $\text{VC}_{\mathcal{X}}(\mathcal{H})$, is the maximal number N of graphs $G_1, \ldots, G_N \subseteq \mathcal{X}$ that can be shattered by $\mathcal{H}$. The idea behind the VC dimension is that although $\mathcal{H}$ may contain infinitely many functions, the different ways it can classify a training set of N sample graphs is finite. Namely, for any given graph in the training sample, a Boolean function can only take the values 0 or 1. On a sample of N graphs $G_1, \ldots, G_N$, a function can act in at most 2^N different ways. Vapnik and Chervonenkis [1964], Vapnik [1998] showed the following bound for the expected error for the 0-1 loss function.

Theorem 1 (Vapnik and Chervonenkis [1964], Vapnik [1998], adapted to MPNNs)**.** Let $\mathcal{H}$ be a hypothesis class of MPNNs over the set of graphs $\mathcal{X}$, with $VC_{\mathcal{X}}(\mathcal{H}) = d < \infty$. Then, for all $\delta \in (0, 1)$, with probability at least $1 - \delta$, the following holds for all MPNN $f \in \mathcal{H}$,

$$\ell_{\text{exp}}(f) \leq \ell_{\text{emp}}(f) + \sqrt{\frac{2d \log(eN/d)}{N}} + \sqrt{\frac{\log(1/\delta)}{2N}}.$$

The proof of the above result follows the following process. First, we use the so-called symmetrization lemma (see [Bousquet et al., 2003, Lemma 2]) to bound $\sup_{f \in \mathcal{H}} |\ell_{\text{emp}}(f) - \ell_{\text{exp}}(f)|$ by a sum of finite terms, utilizing the fact that any hypothesis class $\mathcal{H}$ can only act in at most 2^N different ways on an N-size sample. Then, we apply Hoeffding's inequality [Boucheron et al., 2003] to each term of the finite sum that arises.

It is worth mentioning that Theorem 1 generally holds for any input data space $\mathcal{X}$ beyond graph data and any hypothesis class $\mathcal{H}$. Hence, bounding the VC dimension of a hypothesis class of Boolean concepts directly implies an upper bound on the generalization gap. Note that the analysis is only meaningful in cases where the number of samples is much larger than the VC dimension.

3.1. Bounding the VC dimension of MPNNs

In one of the earliest studies aiming to bound the VC dimension of MPNNs, Scarselli et al. [2018], based on Karpinski and Macintyre [1997], derived bounds for the VC dimension for a specific class of MPNNs. They examined the influence of various factors, e.g., the number of parameters and the maximum number of vertices in the input graph, based on the choice of the

activation function for the utilized feed-forward network, using the framework of Pfaffian function theory [Karpinski and Macintyre, 1997]. It is worth noting that the bounds presented in Scarselli et al. [2018] closely resemble those established for recurrent neural networks. This similarity was not initially expected, given the complexity of MPNN architectures, which could have potentially affected the VC dimension differently. In more recent work, D'Inverno et al. [2024], using similar assumptions on the activation functions and following similar proof techniques as Scarselli et al. [2018], extended the analysis of the VC dimension to modern MPNNs. Since Scarselli et al. [2018] established their bounds for the earliest MPNN model [Scarselli et al., 2009], the architecture and notation in [Scarselli et al., 2018] slightly differs from the one introduced in Section 2.1. Therefore, we choose to present the results from D'Inverno et al. [2024], adapted to the recent and most commonly used MPNN architecture described in Equations (1) and (2).

Specifically, D'Inverno et al. [2024] provided upper bounds for the VC dimension of modern MPNN architectures under the assumption of Pfaffian activation functions; see Appendix B. The bounds depend on the total number of parameters p, the number of layers L, the feature dimension d, the size q of the hidden vector representation of each node v after the Lth layer, and the maximal order n of the graphs.

While Theorem 1 in D'Inverno et al. [2024] provides the analytical expression of the VC bound for MPNNs, we present the following result showing the dependence of the VC dimension on different MPNN architectures and graph parameters. The following result provides bounds for a restricted class $\mathcal{H}$ of MPNNs, where the functions UPD, AGG, and READOUT are Pfaffian functions, and their number of parameters for each layer is in $\mathcal{O}(d)$.

Theorem 2. Let $\mathcal{F}$ be the hypothesis class described above. Then,

$$\mathrm{VC}_{\mathcal{G}_{\leq n,d}^{\mathbb{R}}}(\mathcal{H}) \leq \mathcal{O}(p^{-4}),$$

$$\mathrm{VC}_{\mathcal{G}_{\leq n,d}^{\mathbb{R}}}(\mathcal{H}) \leq \mathcal{O}(n^2),$$

$$\mathrm{VC}_{\mathcal{G}_{\leq n,d}^{\mathbb{R}}}(\mathcal{H}) \leq \mathcal{O}(L^4),$$

$$\mathrm{VC}_{\mathcal{G}_{\leq n,d}^{\mathbb{R}}}(\mathcal{H}) \leq \mathcal{O}(d^6),$$

$$\mathrm{VC}_{\mathcal{G}_{\leq n,d}^{\mathbb{R}}}(\mathcal{H}) \leq \mathcal{O}(q^2),$$

where p is the total number of parameters, L is the depth of MPNNs, q the dimension of the output representation vector, and d the node feature dimension.

Furthermore, D'Inverno et al. [2024] established alternative bounds for the VC dimension of MPNNs based on the color count produced by the 1-WL algorithm. However, we choose not to delve into these findings as we will explore the exact correlation of the 1-WL algorithm, see Appendix A, and the VC dimension in the following section.

3.2. Connecting the 1-WL and VC dimension bounds for MPNNs

Morris et al. [2023] also explored bounds for the VC dimension. What is particularly intriguing about their work is the connection they establish between the VC dimension of a subclass of MPNNs and the 1-WL; see Appendix A. While it was previously known [Morris et al., 2019b, Xu et al., 2019a] that the 1-WL is closely linked with the expressivity of MPNNs, little was understood about its relationship with the generalization capabilities of MPNNs. Their study is divided into deriving upper bounds for the VC dimension of MPNNs under two scenarios.

One where an upper bound on the graph's order is known (*non-uniform*), and one without this assumption (*uniform*). In the former case, leveraging the equivalence between MPNNs and 1-WL [Morris et al., 2019b, Xu et al., 2019a], they demonstrate that the VC dimension of MPNNs with L layers equals the maximum number of graphs distinguishable in L iterations of 1-WL. By doing so, they established the *first lower bound* for the VC dimension of MPNNs. It is essential to note that this result was established only for the case with Boolean vertex features. For scenarios where the graph's order is unbounded, they establish that the number of parameters, specifically the bit length of MPNNs' weights, tightly bounds their VC dimension; see also [Daniëls and Geerts, 2024] for a refined result. Additionally, they provide an upper bound for MPNNs' VC dimension using the number of colors produced by the 1-WL for each graph. We summarize the above results in the next theorems.

Concretely, they consider the following slightly modified definition of VC dimension. For a hypothesis class $\mathcal{H}$ of MPNN concepts and $\mathcal{X} \subset \mathcal{G}$ of graphs, $\mathrm{VC}_{\mathcal{X}}(\mathcal{H})$ is the maximal number N of graphs $G_1, \ldots, G_N$ in $\mathcal{X}$ that can be shattered by $\mathcal{H}$. Here, $G_1, \ldots, G_N$ are *shattered* if for any $\boldsymbol{\tau}$ in $\{0,1\}^N$ there exists an MPNN concept $f \in \mathcal{H}$ such that for all i in $[N]$,

$$f(G_i) \begin{cases} \geq 2/3 & \text{if } \tau_i = 1, \text{ and} \\ \leq 1/3 & \text{if } \tau_i = 0. \end{cases}$$

Note that the above two thresholds are arbitrary, and the below results also work with other constants. Morris et al. [2023] first consider the VC dimension of MPNN concepts on the class $\mathcal{G}_{n,d}^{\mathbb{B}}$ consisting of graphs of an order of at most n with d-dimensional Boolean features. Let $m_{n,d,L}$ be the maximal number of graphs in $\mathcal{G}_{\leq n,d}^{\mathbb{B}}$ distinguishable by 1-WL after L iterations. Then, $m_{n,d,L}$ is also the maximal number of graphs in $\mathcal{G}_{\leq n,dn,d}^{\mathbb{B}}$ that can be shattered by L-layer MPNN concepts $\mathsf{MPNN}(L)$, as is stated next.

Proposition 3. For all n, d and L, it holds that

$$\mathrm{VC}_{\mathcal{G}_{\leq n,d}^{\mathbb{B}}}\big(\mathsf{MPNN}(L)\big) \leq m_{n,d,L}.$$

This upper bound holds regardless of the choice of aggregation, update, and readout functions used in the MPNNs. They next show a matching lower bound for the VC dimension of MPNNs on graphs in $\mathcal{G}_{\leq n,dn,d}^{\mathbb{B}}$.

Proposition 4. For all n, d, and L, all $m_{n,d,L}$ 1-WL-distinguishable graphs of order at most n with d-dimensional Boolean features can be shattered by sufficiently wide L-layer MPNN concepts. Hence,

$$\mathrm{VC}_{\mathcal{G}_{\leq n,d}^{\mathbb{B}}}\big(\mathsf{MPNN}(L)\big) = m_{n,d,L}.$$

The lower bound follows from the fact that MPNNs are as powerful as the 1-WL [Morris et al., 2019a, Xu et al., 2018].

In the uniform case, Morris et al. [2023] considered the class of graphs $\mathcal{G}_{d,\leq u}^{\mathbb{R}}$ having d-dimensional real vertex features and at most u colors under the 1-WL, resulting in the following result.

Theorem 5. For d and $L \in \mathbb{N}$, and MPNN concepts in $\mathsf{MPNN}_{\mathsf{FNN}}(d, L)$, assume we are using piece-wise polynomial activation functions with $p > 0$ pieces and degree $\delta \geq 0$. Let

$P = d(2dL + L + 1) + 1$ be the number of parameters in the MPNNs. For all u in $\mathbb{N}$,

$$\mathrm{VC}_{\mathcal{G}_{d, \leq u}}(\mathsf{MPNN}_{\mathsf{FNN}}(d, L)) \leq \begin{cases} \mathcal{O}(P \log(puP)) & \text{if } \delta = 0, \\ \mathcal{O}(LP \log(puP)) & \text{if } \delta = 1, \\ \mathcal{O}(LP \log(puP) + L^2 P \log(\delta)) & \text{if } \delta > 1. \end{cases}$$

The dependence on u in these upper bounds cannot be improved by more than a constant factor.

3.3. Margin-based VC bounds for MPNNs

Franks et al. [2023] investigated the VC dimension of MPNNs and graph kernels based on the 1-WL from a data margin perspective, building on the theory of VC dimension of partial concepts outlined in Alon et al. [2021]. Formally, let $\mathcal{X}$ be a non-empty set, following Alon et al. [2021], they consider *partial concepts* $\mathcal{H} \subseteq \{0, 1, \star\}^{\mathcal{X}}$, where each concept $c \in \mathcal{H}$ is a *partial function*. That is, if $x \in \mathcal{X}$ such that $c(x) = \star$, then c is *undefined* at x. The *support* of a partial concept $h \in \mathcal{H}$ is the set

$$\mathsf{supp}(h) := \{x \in \mathcal{X} \mid h(x) \neq \star\}.$$

Alon et al. [2021] showed that the VC dimension of (total) concepts straightforwardly generalizes to partial concepts. As with the original definition of VC dimension, the VC dimension of a partial concept class $\mathcal{H}$ over a set of $\mathcal{X}$, denoted $\mathrm{VC}_{\mathcal{X}}(\mathcal{H})$, is the maximum cardinality of a shattered set $U := \{x_1, \dots, x_N\} \subseteq \mathcal{X}$. Here, the set U is *shattered* if for any $\boldsymbol{\tau} \in \{0, 1\}^N$ there exists $f \in \mathcal{H}$ such that for all $i \in [N]$,

$$f(x_i) = \tau_i.$$

In essence, Alon et al. [2021] showed that the standard definition of PAC learnability extends to partial concepts, recovering the equivalence of finite VC dimension and PAC learnability.

Following Alon et al. [2021], a *sample* $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N) \in \mathbb{R}^d \times \{0, 1\}$, for $d > 0$, is (r, λ)-*separable* if (1) there exists $\boldsymbol{p} \in \mathbb{R}^d$, $r > 0$, and a ball $B(\boldsymbol{p}, r)$ such that $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N \in B(\boldsymbol{p}, r)$ and (2) the Euclidean distance between $\mathsf{conv}(\{\boldsymbol{x}_i \mid y_i = 0\})$ and $\mathsf{conv}(\{\boldsymbol{x}_i \mid y_i = 1\})$ is at least 2λ. Then, the sample S is *linearly separable* with *margin* λ. We define the set of concepts

$$\mathcal{H}_{r, \lambda}(\mathbb{R}^d) := \Big\{ h \in \{0, 1, \star\}^{\mathbb{R}^d} \;\Big|\; \forall \boldsymbol{x}_1, \dots, \boldsymbol{x}_N \in \mathsf{supp}(h) :$$
$$(\boldsymbol{x}_1, h(\boldsymbol{x}_1)), \dots, (\boldsymbol{x}_N, h(\boldsymbol{x}_N)) \text{ is } (r, \lambda)\text{-separable} \Big\}.$$

Alon et al. [2021] showed that the VC dimension of the concept class $\mathcal{H}_{r, \lambda}(\mathbb{R}^d)$ is asymptotically lower- and upper-bounded by r^2/λ^2. Importantly, the above bounds are independent of the dimension d, while standard VC dimension bounds scale linearly with d [Anthony and Bartlett, 2002].

Franks et al. [2023] lifted the result to MPNNs. Assuming a fixed but arbitrary number of layers $T \geq 0$, a number of vertices $n > 0$, and an feature dimension $d > 0$, let

$$\mathcal{E}_{\mathsf{MPNN}}(n, d, T) := \{G \mapsto f(G) \mid G \in \mathcal{G}_n \text{ and } f \in \mathsf{MPNN}_{\mathsf{FNN}}(d, T)\},$$

i.e., the set of d-dimensional vectors computable by simple T-layer MPNNs over the set of n-order graphs. A (graph) sample $(G_1, y_1), \dots, (G_N, y_N) \in \mathcal{G}_n \times \{0, 1\}$ is (r, λ)-$\mathcal{E}_{MPNN}(n, d, T)$-*separable*

if there is an MPNN $f \in \mathcal{E}_{\mathsf{MPNN}}(n, d, T)$ such that $(f(G_1), y_1), \ldots, (f(G_N), y_N) \in \mathbb{R}^d \times \{0, 1\}$ is (r, λ)-separable, resulting in the set of partial concepts

$$\mathcal{H}_{r,\lambda}(\mathcal{E}_{\mathsf{MPNN}}(n, d, T)) \coloneqq \Big\{ h \in \{0, 1, \star\}^{\mathcal{G}_n} \;\Big|\; \forall\, G_1, \ldots, G_N \in \mathsf{supp}(h) \colon$$

$$(G_1, h(G_1)), \ldots, (G_N, h(G_N)) \text{ is } (r, \lambda)\text{-}\mathcal{E}_{\mathsf{MPNN}}(n, d, T)\text{-separable} \Big\}.$$

Based on this, Franks et al. [2023] showed that the VC dimension of MPNNs can be tightly bounded by the margin of linearly separable data, resulting in the following theorem.

Theorem 6. For any $n, T > 0$ and sufficiently large $d > 0$, we have,

$$\mathrm{VC}(\mathcal{H})_{\sqrt{T+1}n, \lambda}(\mathcal{E}_{\mathsf{MPNN}}(n, d, T))) \in \Theta(r^2/\lambda^2), \text{ for } r = \sqrt{T}n \text{ and } n \geq r^2/\lambda^2.$$

Moreover, they also showed the same bound for more expressive MPNNs incorporating subgraph information, following Bouritsas et al. [2020]. In addition, they derived conditions for when such more expressive architectures lead to a larger margin, resulting in improved generalization performance.

4. Rademacher complexity bounds

Next, we introduce *Rademacher complexity*, see Mohri et al. [2018], a concept similar to VC dimension, measuring the capacity of a concept class. The fundamental distinction is that while the VC dimension remains independent of the data distribution, the Rademacher complexity relies on the underlying data distribution. Formally, let $\sigma_1, \ldots, \sigma_N$ be independent random variables taking values of -1 and 1 with equal probabilities of 0.5 each (i.e., Rademacher random variables). We define the Rademacher complexity $\mathcal{R}(\mathcal{H})$ of a hypothesis class $\mathcal{H}$ consisting of binary classification concepts, i.e., mappings from $\mathcal{X} \to \{-1, +1\}$, as

$$\mathcal{R}(\mathcal{H}) \coloneqq \mathbb{E}_{\sigma, P_X} \left[\sup_{f \in \mathcal{H}} \left(\frac{1}{n} \sum_{i=1}^{n} \sigma_i f(X_i) \right) \right].$$

Here, σ are the Rademacher random variables, and P_X is the marginal distribution of the initial underlying data distribution P, as described in Section 2.3.1. To understand the above expression, consider the values of σ_i as fixed, where each σ_i represents a label for the data point X_i. Since both σ_i and the concept $f(X_i)$ take values of $+1$ or -1, their product $\sigma_i f(X_i)$ yields $+1$ if σ_i equals $f(X_i)$, and -1 otherwise. Consequently, the summation becomes large when the predicted labels $f(X_i)$ match the σ_i labels across many data points. Hence, the function f fits well with the σ_i labels. Considering the supremum, we examine not just one function f but all functions $f \in \mathcal{H}$. The supremum becomes large if a function in $\mathcal{H}$ fits the given sequence of σ_i labels effectively. As σ_i represent random variables, they serve as random labels for the data points X_i. Thus, we assign a higher Rademacher complexity value to the class $\mathcal{H}$ if it can effectively adapt to random labels, aligning with the understanding that a function space needs to be sufficiently large to accommodate diverse random labels across varied datasets.

Finally, it is worth mentioning that in the definition of the Rademacher complexity, the expectation is computed regarding both Rademacher random variables and the data marginal distribution P_X. This makes it non-computable since the underlying distribution P is unknown. Thereto, we introduce the empirical Rademacher complexity to provide generalization bounds.

The empirical Rademacher complexity differs because the expectation is computed only regarding the Rademacher random variables σ_i, while for the X_i, we use the training set. Therefore, the *empirical Rademacher complexity*

$$\hat{\mathcal{R}}(\mathcal{H}) := \mathbb{E}_\sigma\left[\sup_{f \in \mathcal{H}}\left(\frac{1}{N}\sum_{i=1}^{N}\sigma_i f(X_i)\right)\right].$$

From a mathematical perspective, the (empirical) Rademacher complexity can be employed to bound the generalization error, utilizing the following result [Mohri et al., 2018].

Theorem 7 ([Bousquet et al., 2003][Theorem 5]**.**] Let $\mathcal{H}$ be a hypothesis class mapping from a set of graphs $\mathcal{G}$ (e.g., $\mathcal{G}_{n,d}^\mathbb{R}$) to $\{0,1\}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, over the draw of an i.i.d. sample of size N, the following holds:

$$\ell_{\exp}(f) \leq \ell_{\emp}(f) + 2\mathcal{R}(\mathcal{H}) + \sqrt{\frac{\log(1/\delta)}{2N}}.$$

and also, with probability at least $1 - \delta$,

$$\ell_{\exp}(f) \leq \ell_{\emp}(f) + 2\hat{\mathcal{R}}(\mathcal{H}) + \sqrt{\frac{2\log(2/\delta)}{N}}.$$

The important part of the above result is the second inequality, showing that one can obtain a bound that depends solely on the data. The proof of the above theorem again utilizes the symmetrization lemma and another concentration inequality (McDiarmid's inequality, see [McDiarmid, 1989]) to bound the difference between the Rademacher complexity and the empirical Rademacher complexity.

4.1. Rademacher complexity bounds for MPNNs

Garg et al. [2020] derived data-dependent bounds based on bounding the Rademacher complexity, tailored specifically to MPNN architectures, extending results from Bartlett et al. [2017] for feed-forward neural networks. Their findings show that these bounds are significantly tighter compared to VC dimension-based bounds for MPNNs outlined in Scarselli et al. [2018], and they resemble Rademacher bounds for recurrent neural networks [Chen et al., 2020].

They consider the following popular MPNN architecture [Dai et al., 2016, Jin et al., 2019] for computing vertex features. Let G be a graph, for $v \in V(G)$ with initial feature vector $\boldsymbol{h}_v^{(0)} \in \mathbb{R}^d$, at layer $l > 0$, we set

$$\boldsymbol{h}_v^{(l)} := \phi\left(\boldsymbol{h}_v^{(l-1)}\boldsymbol{W}_1 + \rho\left(\sum_{u \in N(v)} g\left(\boldsymbol{h}_u^{(l-1)}\right)\right)\boldsymbol{W}_2\right) \in \mathbb{R}^{1 \times d},$$

where ϕ, ρ, and g are pointwise nonlinear activation functions and $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{d \times d}$ are learnable parameters of the MPNN. They further assume that $\|\boldsymbol{h}_v^{(0)}\|_2 \leq B$ for $v \in V(G)$, and that ϕ is bounded by a constant b. For example, ϕ could be the hyperbolic tangent function. Furthermore, they assume that ϕ, ρ, and g are Lipschitz continuous functions with constants C_ϕ, C_ρ, and C_g, respectively, and that the weights $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ have bounded norms, i.e., $\|\boldsymbol{W}_i\|_2 \leq B_i$, for $i \in [2]$. Note that in this architecture, the weights $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ and the functions ϕ, ρ, and g

14

are shared across all vertices and layers. Clearly, the above architecture is a special case of the general form of MPNNs described in Equation (1).

The final step in the MPNN computation involves the readout layer, which generates a single feature vector $\boldsymbol{h}_G$ for the entire graph after the Lth layer as in Equation (2), namely,

$$f_c(\boldsymbol{h}_v^{(L)}) := \sigma\left(\boldsymbol{\beta}^\mathsf{T}\boldsymbol{h}_v^{(L)}\right), \text{ for all } v \in V(G), \quad \text{and} \quad \boldsymbol{h}_G := \sum_{v \in V} f_c\left(\boldsymbol{h}_v^{(L)}\right),$$

where $\boldsymbol{h}_v^{(L)}$ is the feature embedding of node v in the last layer L, $\boldsymbol{\beta} \in \mathbb{R}^{d_L}$ is a parameter of the readout function with bounded norm ($\|\boldsymbol{\beta}\|_2 < B_{\boldsymbol{\beta}}$), and σ denotes the sigmoid function. Since we are considering a binary classification problem, we can, without loss of generality, consider labels $\{0,1\}$ instead of $\{-1,+1\}$. The loss function for a graph G with true label y and a concept $f \colon \mathcal{G} \to \{0,1\}$ is given by the margin γ loss function,

$$\text{loss}_\gamma(\alpha) := \mathbf{1}[a > 0] + (1 + \alpha/\gamma)\mathbf{1}[\alpha \in [-\gamma, 0]],$$

where, $\mathbf{1}_A$ denotes the indicator function of an event A, α is defined through the population risk $p(f(G), y) := y(2f(G) - 1) + (y - 1)(2f(G) - 1)$ as $\alpha = -p(f(G), y)$. The margin loss function evaluates the classifier's output by considering whether the prediction is correct and how confident the prediction is. A larger margin indicates higher confidence in the correct classification. See Lin [2004] for a detailed analysis of margin loss functions.

The following result shows the relation between the population risk $\mathbb{P}[p(f(G), y) \leq 0]$, the empirical risk $\ell_{\text{emp}}(f) = \frac{1}{N}\sum_{j=1}^N \text{loss}_\gamma(-p(f(G_i), y_i))$ for the training set $\{(G_i, y_i)\}_{i=1}^N$ and the empiricial Rademacher complexity.

Lemma 8 ([Mohri et al., 2018])**.** For any margin $\gamma > 0$, any prediction function f in a hypothesis class $\mathcal{H}$ and $\mathcal{J}_\gamma \in \{(G, y) \mapsto \text{loss}_\gamma(-p(f(G), y)) \mid f \in \mathcal{H}\}$, given N samples (G_j, y_j) sampled i.i.d. from P, with probability at least $1 - \delta$, the population risk for P and f is bounded as,

$$\mathbb{P}[p(f(G), y) \leq 0] \leq \ell_{\text{emp}}(f) + 2\hat{\mathcal{R}}(\mathcal{J}_\gamma) + 3\sqrt{\frac{\log(2/\delta)}{2N}},$$

where $\hat{\mathcal{R}}(\mathcal{J}_\gamma)$ is the empirical Rademacher complexity of the hypothesis class $\mathcal{J}_\gamma$.

Therefore, to bound the population risk, Garg et al. [2020] derived bounds for the empirical Rademacher complexity. These bounds were derived in two steps. Firstly, they demonstrated that bounding the empirical Rademacher complexity can be achieved by bounding the Rademacher complexity of vertex-wise computation trees. Then, they bounded the complexity of a single computation tree via recursive spectral bounds, resulting in the following proposition.

Proposition 9. If the maximum degree for any node is at most $\tilde{d}$, then the empirical Rademacher complexity $\hat{\mathcal{R}}(\mathcal{J}_\gamma)$, is bounded by,

$$\hat{\mathcal{R}}(\mathcal{J}_\gamma) \leq \frac{4}{\gamma N} + \frac{24 d B_{\boldsymbol{\beta}} Z}{\gamma\sqrt{N}}\sqrt{3\log Q}, \text{ where}$$
$$Q := 24 B_{\boldsymbol{\beta}}\sqrt{N}\max\{Z, M\sqrt{d}\max\{B_h B_1, \bar{R}B_2\}\},$$
$$M := C_\phi\frac{(C\tilde{d})^L - 1}{C\tilde{d} - 1}, \ Z := C_\phi B_1 B_h + C_\phi B_2 \bar{R},$$
$$C := C_\rho C_g C_\phi B_2,$$
$$\bar{R} \leq C_\rho C_g \tilde{d}\min\{b\sqrt{d}, B_1 B_h M\}.$$

The most important parameters in the above result are the dimension of the feature space d, the size N of the training data, and the number of layers L. All other constants in the bound stem from the model's parameters and the Lipschitz constants of the non-linear activation functions. These bounds exhibit the same dependence on dimension d, depth L, and sample size N as the generalization bounds established for recurrent neural networks [Chen et al., 2020]. Additionally, the VC bounds established by Scarselli et al. [2018] for the setting with tanh and sigmoid activation functions depend on the fourth order regarding the number of hidden units and quadratic in dimension d and the maximum number of nodes n in the input graph. However, in the above setting, the VC dimension scales as $\mathcal{O}(d^6 n^2)$ and consequently, the generalization error as $\mathcal{O}(d^3 n/\sqrt{N})$. Thus, the bounds established by Garg et al. [2020] are significantly tighter. Recently, Karczewski et al. [2024] extended the above analysis to equivariant MPNNs.

Rademacher complexity bounds for GCNs Lv [2021] studied the generalization power of single-layer GCN, providing tight upper bounds of Rademacher complexity for GCN models with a single hidden layer. Since these results refer to node-level tasks, as will be clarified, to define the Rademacher complexity in this work, it is assumed that pairs of vertices and labels have been sampled in an i.i.d. fashion according to some underlying distribution P. The Rademacher bounds derived in this work explicitly depend on the largest eigenvalue of the graph (convolution) filter and the degree distribution of the graph. Another important difference from the work by Garg et al. [2020] is that only the vertex-focused task involving a fixed adjacency matrix is considered in this study. Therefore, the data is no longer subject to the i.i.d. assumptions. Furthermore, they provide a lower bound of the Rademacher complexity of the aforementioned models, demonstrating the optimality of their derived upper bound.

We consider an undirected graph $G \in \mathcal{G}_{n,d}$ with $V(G) = [n]$ and $\boldsymbol{A}, \boldsymbol{L}$ being the adjacency matrix of G with added self-connections and the Laplacian matrix, respectively. We denote by $\Omega \subset V(G)$ the set of vertex indices with observed labels such that $m := |\Omega| < n$. Additionally, for $d \in \mathbb{N}$, an each vertex $v \in \Omega$, we denote by $\boldsymbol{h}_v \in \mathbb{R}^d$ its feature vector and its label by $y_v \in \mathcal{Y} \subset \mathbb{R}$, while for each $u \in V(G) \setminus \Omega$ we only know the feature vectors in $\mathbb{R}^d$. The goal is to predict y_u for every $u \in V(G) \setminus \Omega$. We consider the single-layer GCN architecture as described in Equation (3).

The predicted label of vertex v is denoted as $f_G(v)$. The expected loss is therefore defined as

$$\ell_{\exp}(f_G) := \mathbb{E}_P[\ell(f_G, v, y],$$

where ℓ is the loss function and the distribution P is the underlying distribution on $\mathbb{R}^d \times \mathcal{Y}$. Similarly, given a training set through the node indices Ω as previously described, we can compute the empirical risk

$$\ell_{\emp}(f_G) := \frac{1}{N} \sum_{u \in \Omega} \ell(f_G, u, y_i).$$

Now, consider the class $\mathcal{F}_{D,R}$ to be the class of two-layer graph neural networks over $\mathbb{R}^d$, under some technical assumptions analytically described in the original paper. The constants D and R are also defined according to these assumptions. Especially, D bounds the L_2 norm of the weight vector of the output ($\|\boldsymbol{w}^{(2)}\|_2 \leq D$), and R bounds the Frobenius norm of the weight matrix in the first layer ($\|\boldsymbol{W}\|_F^{(1)} < R$). Finally, an important assumption is that they consider all vertex having degree q (e.g., regular graphs). Let $\hat{\mathcal{R}}(\mathcal{F}_{D,R})$ be the empirical Rademacher complexity

of class $\mathcal{F}_{D,R}$ and let $\mathcal{F} := \{\ell(y, f(\cdot)) : f \in \mathcal{F}_{D,R}\}$ where ℓ is a α_ℓ-Lipschitz continuous function. By Bartlett and Mendelson [2001], we have that

$$\ell_{\exp}(f) \leq \ell_{\mathrm{emp}}(f) + 2\hat{\mathcal{R}}(\mathcal{F}) + \sqrt{\frac{2\log(2/\delta)}{N}}, \text{ for all } f \in \mathcal{F},$$

and also $\hat{\mathcal{R}}(\mathcal{F}) \leq 2\alpha_\ell\hat{\mathcal{R}}(\mathcal{F}_{D,R})$.

Combining the above two results and bounding the empirical Rademacher complexity of $\mathcal{F}_{D,R}$, Lv [2021] proved the following generalization bound.

Theorem 10. Let f be any given predictor of a class of GCNs with one hidden layer on a graph where all vertex degree q. For any $\delta > 0$, with probability at least $1 - \delta$, we have

$$\ell_{\exp}(f) \leq \ell_{\mathrm{emp}}(f) + 16M^2BDR\alpha_\ell|\lambda_{\max}(G)|\sqrt{\frac{q}{m}}\sum_{l=1}^{q}\max_{j\in[m]}|[g(\boldsymbol{L})]_{jn_l(j)}| + \sqrt{\frac{2\log(2/\delta)}{n}},$$

where M is the Lipschitz constant of the activation function σ, B bounds the L_2-norm of feature vectors in the input space, $\lambda_{\max}$ is the maximum absolute eigenvalue of the graph filter $g(\boldsymbol{L})$, and $n_l(j)$ refers to the l-th neighbor of node v_j by a given order.

5. Stability-based generalization bounds

So far, we have only seen generalization bounds based on VC dimension theory and the Rademacher complexity of a hypothesis class. In all previous approaches, the bounds were based on bounding the capacity of the set of functions that the learning machine can implement. As nicely observed and described in von Luxburg and Schölkopf [2011], all the above bounds have the following form,

$$\ell_{\exp}(f) \leq \ell_{\mathrm{emp}}(f) + \mathrm{capacity}(\mathcal{H}) + \mathrm{confidence}(\delta), \text{ for all } f \in \mathcal{H}.$$

In this context, $\mathcal{H}$ represents our hypothesis class, while the confidence term depends on the probability with which the bound should hold. It is important to recognize that all bounds structured this way are inherently pessimistic, as they apply universally to every possible learning algorithm that can be used to choose a function from $\mathcal{H}$. Consequently, the most extreme or problematic learning algorithm influences the bound's behavior. However, typically, real-world optimizers do not tend to pick the worst function in a model class.

Verma and Zhang [2019], based on Bousquet and Elisseeff [2002], followed a different approach to derive generalization bounds for single-layer GCN. Instead of uniformly bounding the capacity of the hypothesis class, they derive generalization bounds for learning algorithms satisfying useful properties. They also consider the inherent randomness in estimating the MPNN concept from sample data in the following setting. Intuitively, this randomness arises during the optimization of the MPNN concept using stochastic gradient descent. We first recall the notion of a learning algorithm, which is the optimization process for estimating the MPNN concept.

For an input space $\mathcal{X}$, an output space $\mathcal{Y}$, and a hypothesis class $\mathcal{H}$, a learning algorithm $\mathcal{A}$ maps an N-size sample from $\mathcal{Z}^N$ to a concept $\mathcal{H}$, where $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Given the training set $\mathcal{S}$ as $\mathcal{A}_{\mathcal{S}}$, we denote the concept learned. Here, $\mathcal{A}_{\mathcal{S}}$ represents an MPNN concept. Below, we define *uniform stability* for a randomized algorithm. Let $\mathcal{S} = \{z_1, z_2, \ldots, z_N\} \in \mathcal{Z}^N$ denote an N-sized sample. Additionally, we define $\mathcal{S}^{\setminus i} = \mathcal{S} \setminus \{z_i\}$, for $i \in [N]$.

Definition 11. For a given sample size $N \in \mathbb{N}$, a randomized learning algorithm $\mathcal{A}$ is β-*uniformly stable* with respect to a loss function ℓ if it satisfies

$$\sup_{\mathcal{S} \in \mathcal{Z}^N, z \in \mathcal{Z}} \{|\mathbb{E}_{\mathcal{A}}[\ell(\mathcal{A}_{\mathcal{S}}, z)] - \mathbb{E}_{\mathcal{A}}[\ell(\mathcal{A}_{\mathcal{S} \backslash i}, z)]|\} \le 2\beta, \quad \text{for all } i \in [N].$$

Here, the expectation regarding $\mathcal{A}$ is over the implicit randomness in the algorithm. In our scenario, we can regard the algorithm $\mathcal{A}$ as the optimization method used during the learning process (e.g., SGD).

Based on the stability properties of single-layer GCN models, Verma and Zhang [2019] derived generalization bounds for node classification tasks under the assumption that the given graph can be decomposed into distinct 1-hop subgraphs, each corresponding to the 1-neighborhood of an individual node in the original graph. They further assume that these decomposed subgraphs are sampled independently and identically from an underlying distribution.

More precisely, following Bousquet and Elisseeff [2002, Theorem 12], they first utilized the algorithmic uniform stability property to derive generalization error bounds in terms of the uniform stability parameter [Verma and Zhang, 2019, Theorem 2]. Subsequently, they showed that the algorithmic stability parameter is bounded by the largest absolute eigenvalue of the graph convolution filter [Verma and Zhang, 2019, Theorem 3]. By combining these results, they established the following generalization bound as a function of the largest eigenvalue of the graph convolution filter.

Theorem 12 (Verma and Zhang [2019], Theorem 1)**.** Let $\mathcal{A}_{\mathcal{S}}$ be a single-layer GCN equipped with the graph convolution filter $g(\boldsymbol{L})$ and trained on a dataset S using the SGD algorithm for T iterations. Assume the loss and activation functions are Lipschitz continuous and smooth. Then, for all $\delta \in (0, 1)$ with probability at least $1 - \delta$,

$$\mathbb{E}_{\text{SGD}}[\ell_{\exp}(\mathcal{A}_{\mathcal{S}}) - \ell_{\text{emp}}(\mathcal{A}_{\mathcal{S}})] \le \frac{1}{n}\mathcal{O}\big((\lambda_G^{\max})^{2T}\big) + \Big(\mathcal{O}\big((\lambda_G^{\max})^{2T}\big) + M\Big)\sqrt{\frac{\log(1/\delta)}{2N}},$$

where $\lambda_G^{\max}$ is the absolute largest eigenvalue of $g(\boldsymbol{L})$, N represents the number of training samples, and M stands for a constant depending on the loss function.

Building on the work of Bousquet and Elisseeff [2002], Mukherjee et al. [2006], Zhou and Wang [2021] established stability-based generalization bounds dependent on graph filters and their product with node features for graph convolutional networks with multiple layers. They demonstrated that the generalization error of GCNs tends to grow with more layers, providing an explanation for why GCNs with deeper layers exhibit relatively poorer performance on test datasets.

6. PAC-Bayesian bounds

In this section, we present the PAC-Bayesian approach [McAllester, 1999, 2003, Langford and Shawe-Taylor, 2002], which is an alternative framework for deriving generalization bounds by assuming a prior distribution of the hypothesis class instead of having a deterministic model as in common learning formulations.

We slightly modify the previous notation here. For multiclass classification, $f_{\boldsymbol{\omega}}$ is a function from the hypothesis class, i.e., $f_{\boldsymbol{\omega}} \in \mathcal{H} \subseteq \{f : \mathcal{G}_d \to \mathbb{R}^K\}$, where $\boldsymbol{\omega}$ is the vectorization of all

model parameters and K the number of classes for the multiclass classification problem. For a given loss function $\ell\colon \mathcal{H} \times \mathcal{G}_d \times \mathbb{R}^K \to \mathbb{R}$, as previously, we define the expected and empirical loss of the function $f_{\boldsymbol{\omega}}$ as follows,

$$\ell_{\exp}(f_{\boldsymbol{\omega}}) := \mathbb{E}_{(G,y)\sim P}[\ell(f_{\boldsymbol{\omega}}, G, y)] \quad \text{and} \quad \ell_{\text{emp}}(f_{\boldsymbol{\omega}}) := \frac{1}{N}\sum_{i=1}^{N} \ell(f_{\boldsymbol{\omega}}, X_i, y_i).$$

Here, N is the total number of i.i.d. observations $(G_i, y_i) \sim P$.

In the PAC-Bayes approach, we assume a prior distribution π over the hypothesis class $\mathcal{H}$, and we obtain a posterior distribution Q over the same support through the learning process. Under this Bayesian view, we define the expected and the empirical error for a distribution Q over $\mathcal{H}$, respectively, as

$$L_{\exp}(Q) := \mathbb{E}_{\boldsymbol{\omega}\sim Q}[\ell_{\exp}(f_{\boldsymbol{\omega}})],$$

and

$$L_{\text{emp}}(Q) := \mathbb{E}_{\boldsymbol{\omega}\sim Q}[\ell_{\text{emp}}(f_{\boldsymbol{\omega}})].$$

Following the above notation, McAllester [2003] provided a general tool for deriving generalization bounds for any pair of distributions (π, Q) based on the "distance" between π and Q. This result is obtained using tools from information theory, which leverage the *Kullback–Leibler divergence* (KL) divergence to measure the discrepancy between the prior and posterior distributions. Formally, we have the following theorem.

Theorem 13. For any prior distribution π over $\mathcal{H}$ and any $\delta \in (0,1)$, with probability at least $1-\delta$, the following bound holds for all distributions Q over $\mathcal{H}$,

$$L_{\exp}(Q) \leq L_{\text{emp}}(Q) + \sqrt{\frac{D_{\text{KL}}(Q \parallel \pi) + \log(2N/\delta)}{2(N-1)}}.$$

Here, $D_{\text{KL}}(Q \parallel \pi) := \sum_{x\in\mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$ denotes the Kullback–Leibler divergence, which measures the discrepancy between the distributions Q and π. Specifically, it quantifies the information loss incurred when using Q as an approximation of π. A lower KL divergence indicates that Q is closer to π, while a higher value suggests significant deviation.

While the above theorem provides a theoretical framework for generalization bounds, its direct application remains unclear, as it does not specify which choices of π and Q lead to a "good" bound. The key advantage of this theorem lies in its flexibility, allowing us to select a prior π and a posterior Q that yield a meaningful and computable bound. Therefore, as we will discuss later, a careful selection of π and Q is essential to ensure that the resulting bound is both tight and practically useful.

6.1. PAC-Bayesian bounds for MPNNs and GCNs

Liao et al. [2021], following a PAC-Bayesian approach, established generalization bounds for multilayer GCNs and MPNNs. Their generalization bounds for both architectures depend on the maximum node degree of the graphs and the spectral norm of the weights of each model. Their bounds for GCNs can be seen as a natural generalization of the results developed in Neyshabur et al. [2018] for fully-connected and convolutional neural networks, while for MPNNs, the PAC-Bayesian bounds improve over the Rademacher complexity bounds by Garg et al. [2020].

We focus on the K-class classification problem and make the following assumptions.

(A1) For both MPNNs and GCNs, the maximum hidden dimension across all layers is denoted as h.

(A2) The node features of every node on each graph are contained in a ℓ_2-ball with radius B.

(A3) The maximum node degree in all graphs is $\tilde{d} - 1$.

In the following, we use the multi-class γ-margin loss for some $\gamma > 0$ as our loss function. That is, the expected loss is defined as,

$$\ell_{\exp,\gamma}(f_{\boldsymbol{\omega}}) := \mathbb{P}\Big(f_{\boldsymbol{\omega}}(G)[y] \leq \gamma + \max_{j \neq y} f_{\boldsymbol{\omega}}(G)[j] \Big),$$

and the empirical loss,

$$\ell_{\mathrm{emp},\gamma}(f_{\boldsymbol{\omega}}) := \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\Big(f_{\boldsymbol{\omega}}(G_i)[y_i] \leq \gamma + \max_{j \neq y} f_{\boldsymbol{\omega}}(G_i)[j] \Big),$$

where, $f_{\boldsymbol{\omega}}(G)[j]$ is the j-th entry of $f_{\boldsymbol{\omega}}(G) \in \mathbb{R}^K$, and y_i is the label of G_i. Before presenting the main results of Liao et al. [2021], it is crucial to note that in this approach, the PAC-Bayes framework serves only as a tool to derive a generalization bound, and we do not assume randomness in $\boldsymbol{\omega}$. Hence, the objective is to establish a bound applicable to every deterministic $\boldsymbol{\omega}$. Nonetheless, to employ PAC-Bayes theory, we extend the deterministic $\boldsymbol{\omega}$ to a distribution by introducing a random perturbation u. Moreover, the prior distribution π on the hypothesis class must remain independent of $\boldsymbol{\omega}$. Since π cannot rely on $\boldsymbol{\omega}$, we aim to minimize the Kullback-Leibler divergence between π and Q (where Q is defined by $\boldsymbol{\omega} + u$) by selecting a prior π and a random perturbation u with sufficiently high variance allowing π and $\boldsymbol{\omega} + u$ being similar. Finally, Neyshabur et al. [2018], combining the above idea with Theorem 13, proved the following Lemma, which converts PAC-Bayes bounds to generalization bounds on deterministic $\boldsymbol{\omega}$ for the margin loss.

Lemma 14 (Neyshabur et al. [2018]). Let $f_{\boldsymbol{\omega}} : \mathcal{X} \to \mathbb{R}^K$ be any model with parameters $\boldsymbol{\omega}$, and let π be any prior distribution on the parameters that is independent of the training data. For any $\boldsymbol{\omega}$, we construct a posterior $Q(\boldsymbol{\omega} + u)$ defined by the perturbation u to $\boldsymbol{\omega}$, s.t., $\mathbb{P}(\max_{x \in \mathcal{X}} |f_{\boldsymbol{\omega}+u}(x) - f_{\boldsymbol{\omega}}(x)|_{\infty} < \gamma/4) > 1/2$. Then, for any $\gamma, \delta > 0$, with probability at least $1 - \delta$ over an i.i.d. size-N training set, for any $\boldsymbol{\omega}$, we have,

$$\ell_{\exp,0}(f_{\boldsymbol{\omega}}) \leq \ell_{\mathrm{emp},\gamma}(f_{\boldsymbol{\omega}}) + \sqrt{\frac{2D_{\mathrm{KL}}(Q(\boldsymbol{\omega} + u) \,\|\, \pi) + \log\left(8N/\delta\right)}{2(N-1)}}.$$

By assuming spectral norm bounds on all layers, we can constrain the change in the value of $f_{\boldsymbol{\omega}}$ as $\boldsymbol{\omega}$ is perturbed, thereby controlling $|f_{\boldsymbol{\omega}}(x) - f_{\boldsymbol{\omega}+u}(x)|$ independently of x and enabling the utilization of Lemma 14.

Theorem 15 (GCN generalization bound). Let $f_{\boldsymbol{\omega}} \in \mathcal{H} \subset \{h : \mathcal{G} \to \mathbb{R}^K\}$ be an L-layer GCN under the assumptions (A1)-(A3). Then, for any $\delta, \gamma > 0$, with probability at least $1 - \delta$ over the choice of an i.i.d. size N training set, for any $\boldsymbol{\omega}$, we have,

$$\ell_{\exp,0}(f_{\boldsymbol{\omega}}) \leq \ell_{\mathrm{emp},\gamma}(f_{\boldsymbol{\omega}})$$
$$+ \mathcal{O}\left(\sqrt{\frac{B^2 \tilde{d}^{L-1} L^2 h \log\left(Lh\right) \prod_{i=1}^{L} \| \boldsymbol{W}_i \|_2^2 \sum_{i=1}^{L} (\| \boldsymbol{W}_i \|_F^2 \,/\, \| \boldsymbol{W}_i \|_2^2) + \log\left(NL/\delta\right)}{\gamma^2 N}} \right),$$

where $\tilde{d}, h$ as defined in assumptions (A1)-(A3), and $\boldsymbol{W}_i$, are the parameter matrices of the GCN, as described in Appendix C.

Theorem 16 (MPNN generalization bound). *Let $f_{\boldsymbol{\omega}} \in \mathcal{H} \subset \{h \colon \mathcal{G} \to \mathbb{R}^K\}$ be an L-layer MPNN under the assumptions (A1)-(A3). Then, for any $\delta, \gamma > 0$, with probability at least $1 - \delta$ over the choice of an i.i.d. size N training set, for any $\boldsymbol{\omega}$, we have,*

$$
\ell_{\mathrm{exp},0}(f_{\boldsymbol{\omega}}) \leq \ell_{\mathrm{emp},\gamma}(f_{\boldsymbol{\omega}})
$$
$$
+ \mathcal{O}\left(\sqrt{\frac{B^2\big(\max(\zeta^{-(L+1)}, (\lambda\xi)^{(L+1)/L})\big)^2 L^2 h \log(Lh) \parallel \boldsymbol{\omega} \parallel_2^2 + \log\left(N(L+1)/\delta\right)}{\gamma^2 N}} \right),
$$

where ζ, λ depend on the spectral norm of the parameter matrices of the MPNN, $\|\boldsymbol{\omega}\|_2^2$ depends on the Forbenius norm of the parameter matrices, and ξ is a constant depending on the Lipschitz constants of the activation functions and the maximum node degree $\tilde{d} - 1$.

Comparing the bounds established by Liao et al. [2019] to those in Garg et al. [2020], Scarselli et al. [2018] in terms of their dependency on the maximum node degree and maximum hidden dimension, we can derive the following conclusions. Firstly, the bound proposed by Garg et al. [2020] scales as $\mathcal{O}(d^{L-1}\sqrt{p\log(d^{2L-3})})$, whereas Liao et al. [2019] scales as $\mathcal{O}(d^{L-1})$, indicating a slight improvement. Additionally, the bound by Liao et al. [2019] scales as $\mathcal{O}(\sqrt{h\log h})$, tighter than the Rademacher complexity bound $\mathcal{O}(h\sqrt{\log h})$ by Garg et al. [2020] and the VC-dimension bound $\mathcal{O}(h^4)$ by Scarselli et al. [2018].

Sun and Lin [2024], in a similar work to Liao et al. [2021], extended the establishment of generalization bounds for GCNs and MPNNs using a PAC-Bayesian framework. They also established bounds depending on the spectral norm of the diffusion matrix and the spectral norm of the weights. Specifically, they improve upon the bounds proposed by Liao et al. [2021] by eliminating the exponential dependence on the maximum node degree. Additionally, Wu et al. [2023b] derived non-trivial extensions for the above generalization bounds on GCNs avoiding the exponential dependence on the maximum node degree. Finally, in a recent work, Lee et al. [2024] applied a PAC-Bayesian approach to derive generalization bounds for knowledge graph representation learning, providing a first theoretical contribution in this domain.

6.2. Graph diffusion matrix dependent bound

Ju et al. [2023] followed a PAC Bayesian approach and measured the stability of GNNs against noise perturbations through the Hessian matrices. They derived generalization bounds for a more general architecture of GNNs, including message passing graph neural networks [Gilmer et al., 2017], graph convolution networks [Kipf and Welling, 2017], and graph isomorphism networks [Xu et al., 2019b].

The main difference from the previously described PAC-Bayesian bounds is that in this work, the generalization bounds scale with the largest singular value of the GNN feature diffusion matrix (e.g., the adjacency matrix) instead of the maximum node degree. In their work, they also constructed a lower bound of the generalization gap, which asymptotically matches the upper bound. In the following, we briefly present their framework considering a very general set of GNN models for graph-level prediction tasks, i.e., Dai et al. [2016], Garg et al. [2020], Gilmer et al. [2017], Jin et al. [2018], as well as the derived generalization bounds.

Since in the main result, the generalization bound depends on the graph diffusion matrix, we present the matrix notation of GNNs described in Ju et al. [2023]. Let L be the number of layers (the L-th layer is the pooling layer), and d_t denotes the width of each layer for $t \in [L]$. We use ϕ_t, ρ_t, ψ_t as nonlinear functions centered at zero. Additionally, we assume weight matrices $\boldsymbol{W}^{(t)}$ of dimension $d_{t-1} \times d_t$ for transforming neighboring nodes and another weight matrix $\boldsymbol{U}^{(t)}$ with the same dimension. For the first $L-1$ layers, the node embeddings are recursively computed from the input feature matrix $\boldsymbol{H}^{(0)} := \boldsymbol{X} \in \mathbb{R}^{n \times d}$ for any graph $G \in \mathcal{G}_{n,d}$ with diffusion matrix $\boldsymbol{P}_G$ as follows,

$$\boldsymbol{H}^{(t)} := \phi_t \Big(\boldsymbol{X}\boldsymbol{U}^{(t)} + \rho_t \Big(\boldsymbol{P}_G \psi_t \Big(\boldsymbol{H}^{(t-1)} \Big) \Big) \boldsymbol{W}^{(t)} \Big).$$

For the last layer L, we aggregate using the vector $\mathbf{1}_n$ containing ones everywhere as,

$$H^{(L)} := \frac{1}{n} \mathbf{1}_n^T \boldsymbol{H}^{(L-1)} \boldsymbol{W}^{(L)}.$$

Possible common choices for the graph diffusion matrix $\boldsymbol{P}_G$ include the adjacency matrix $\boldsymbol{A}(G)$ or the normalized adjacency $\boldsymbol{D}(G)^{-1}A(G)$. Note that the above architecture can be considered a special case of the broader MPNN architecture described in Equation (1). We arrive at the following bound for the generalization error.

Theorem 17. Suppose all of the nonlinear functions ϕ_t, ρ_t, ψ_t, and the loss function $\ell(\cdot, y)$ are twice-differentiable, Lipschitz-continuous, and their first and second-order derivatives are both Lipschitz continuous. With probability at least $1 - \delta$, for any $\delta > 0$ and any $\epsilon > 0$, any model f following the previously described architecture satisfies,

$$\ell_{\exp}(f) \le \ell_{\text{emp}}(f) + \mathcal{O}\left(\frac{\log(\delta^{-1})}{N^{\frac{3}{4}}} \right) + \sum_{i=1}^{L} \sqrt{ \left(\frac{CBd_i(\max\|X\|^2\|\boldsymbol{P}_G\|^{2(L-1)})\left(r_i^2 \prod_{j=1}^l s_j^2\right)}{N} \right) },$$

where N is the size of the i.i.d. sample, B is an upper bound of the loss function ℓ, and C is a constant depending on the Lipschitz constants of ϕ_t, ρ_t, ψ_t, and $\ell(\cdot, y)$. The constants r_i, s_j are constants used for bounding the spectral and the Forbenius norm of the weight matrices $\boldsymbol{W}^{(t)}$ and $\boldsymbol{U}^{(t)}$.

The bounds proposed in Ju et al. [2023] scale with the spectral norm of the graph diffusion matrix P_G, while the bounds established in [Garg et al., 2020, Liao et al., 2019] scale with the graph's maximum degree. By utilizing appropriate diffusion matrices [Hamilton et al., 2017, Feng et al., 2022], it can be shown that the spectral norm of $\boldsymbol{P}_G$ is strictly smaller than the maximum degree of the graph, suggesting that the bounds presented in Ju et al. [2023] in many cases might be tighter.

7. Covering number-based generalization bounds

In this section, we present generalization bounds for MPNNs by endowing the space of graphs with a (pseudo-)metric that satisfies two key conditions.

1. MPNNs are Lipschitz-continuous functions concerning this (pseudo-)metric.

2. The induced (pseudo-)metric space of graphs is compact.

If these conditions hold, then bounding the covering number of this space leads to a generalization bound for MPNNs. Before presenting the two main theorems that motivate using graph-metric spaces for deriving generalization bounds, we introduce some notations for covering numbers and metric spaces.

Pseudo-metric spaces and covering numbers Let $\mathcal{X}$ be an arbitrary set. A function $d\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ is called a *pseudo-metric* on $\mathcal{X}$ if it satisfies: (i) $d(x,x) = 0$, (ii) $d(x,y) = d(y,x)$ for all $x, y \in \mathcal{X}$, and (iii) $d(x,y) \leq d(x,z) + d(z,y)$, for all $x, y, z \in \mathcal{X}$. The pair $(\mathcal{X}, d)$ is called a *pseudo-metric space*. If, in addition, $d(x,y) = 0 \Rightarrow x = y$ for all $x, y \in \mathcal{X}$, then d is a *metric*, and $(\mathcal{X}, d)$ is called a *metric space*.

Given a pseudo-metric space $(\mathcal{X}, d)$ and $\varepsilon > 0$, an ε-*cover* of $\mathcal{X}$ is a subset $C \subseteq \mathcal{X}$ such that for every $x \in \mathcal{X}$, there exists a $y \in C$ satisfying $d(x,y) \leq \varepsilon$. The *covering number* of $\mathcal{X}$ is then defined as

$$\mathcal{N}(\mathcal{X}, d, \varepsilon) \coloneqq \min\{m \mid \text{there exists an } \varepsilon\text{-cover of } \mathcal{X} \text{ with cardinality } m\},$$

i.e., the smallest number of elements required to form an ε-cover of $\mathcal{X}$ concerning d. Given a pseudo-metric space $(\mathcal{X}, d)$, one can construct a corresponding metric space $\tilde{\mathcal{X}}$ by considering the equivalence relation $\sim$ on $\mathcal{X}$, where $x \sim y$ if and only if $d(x,y) = 0$. The quotient space $\tilde{\mathcal{X}} = \mathcal{X}/_\sim = \{[x] \mid x \in \mathcal{X}\}$ is then endowed with the metric

$$\tilde{d}([x], [y]) = d(x, y).$$

Finally, given a pseudo-metric space $(\mathcal{X}, d)$, we say that a subset $U \subset \mathcal{X}$ is *open* if for every $x \in U$, there exists $\varepsilon > 0$ such that for all $y \in \mathcal{X}$ with $d(x,y) < \varepsilon$, we have $y \in U$. A pseudo-metric space $(\mathcal{X}, d)$ is *compact* if for every $\varepsilon > 0$ and every ε-cover $\cup_{i \in I} U_i$ of $\mathcal{X}$ consisting of open sets $U_i \subset \mathcal{X}$, there exists a finite subcover $\cup_{i \in J} U_i$ with $J \subset I$, and $|J| < \infty$. It follows that for any compact metric space $(\mathcal{X}, d)$, we have $\mathcal{N}(\mathcal{X}, d, \varepsilon) < \infty$ for all $\varepsilon > 0$. Moreover, if the completion,[3] of a pseudo-metric space is compact, the pseudo-metric space has finite covering.

Covering number-based generalization bounds We now present two closely related theorems that justify using compact metric spaces for deriving generalization bounds in graph learning. The following theorem from Levie [2023, Theorem G.3] is an extension of Maskey et al. [2022, Lemma 2].

Theorem 18. Let $(\mathcal{Z}, d)$ be a metric space with finite covering, μ a Borel distribution on $\mathcal{Z}$, and $\{Z_i\}_{i=1}^N$ be i.i.d. samples from μ for some $N \in \mathbb{N}$. Suppose $\mathcal{H} \subset \{h\colon \mathcal{Z} \to \mathbb{R}\}$ consists of bounded and L-Lipschitz functions for some $L \in \mathbb{R}$. Then, the following holds for any $\delta > 0$, with probability at least $1 - \delta$. For every $f \in \mathcal{H}$,

$$\left| \frac{1}{N} \sum_{i=1}^N f(Z_i) - \mathbb{E}_\mu[f(Z_1)] \right| \leq 2\xi^{-1}(N)L + \frac{1}{\sqrt{2}}\xi^{-1}(N)M(1 + \sqrt{\log(^2/\delta)}),$$

where $\mathbb{E}_\mu[f(Z_1)] \coloneqq \int f(x)\,\mu(dx)$, $\xi(\epsilon) \coloneqq \frac{2\kappa(\epsilon)^2 \log(\kappa(\epsilon))}{\epsilon^2}$, $\kappa(\epsilon) \coloneqq \mathcal{N}(\mathcal{Z}, d, \epsilon)$, and ξ^{-1} is the inverse function of ξ.

---

[3]For any pseudo-metric space $\mathcal{X}$ there is unique pseudo-metric space $\overline{\mathcal{X}}$, called the *completion* of $\mathcal{X}$, in which every Cauchy sequence converges, and such that $\mathcal{X}$ is a dense subset of $\overline{\mathcal{X}}$. An example is $\mathbb{R}$ being the completion of the rationals $\mathbb{Q}$ with the metric $|x - y|$.

This theorem provides a method for deriving generalization bounds given a data distribution over a compact metric space, with a proof based on Hoeffding's inequality. It is the key idea behind the approaches taken by Levie [2023], Rauchwerger et al. [2024]. An alternative similar framework, used by Vasileiou et al. [2024], involves a learning algorithm's notion of *robustness*.

Definition 19. We say that a (graph) learning algorithm for a hypothesis class $\mathcal{H}$, is $(K, \varepsilon(\cdot))$-*robust* if $\mathcal{Z}$ can be partitioned into $K > 0$ sets, $\{C_i\}_{i=1}^K$, such that for all samples $\mathcal{S}$ and $(G, y) \in \mathcal{S}$, the following holds. If $(G, y) \in C_i$, for some $i \in \{1, \ldots, K\}$, then for all $(G', y') \in C_i$,

$$\left| \ell(h_{\mathcal{S}}(G), y) - \ell(h_{\mathcal{S}}(G'), y') \right| < \varepsilon(\mathcal{S}).$$

Recall that $h_{\mathcal{S}}$ is the function returned by the learning algorithm regarding the data sample $\mathcal{S}$, and ℓ is a loss function.

Intuitively, the above definition requires that the difference of the losses of two data points in the same cell is small. Xu and Mannor [2012] showed that a $(K, \epsilon(\cdot))$-*robust* learning algorithm implies a bound on the generalization error.

Theorem 20 (Xu and Mannor [2012], Theorem 3)**.** If we have a $(K, \epsilon(\cdot))$-robust (graph) learning algorithm for $\mathcal{H}$ on $\mathcal{Z}$, then for all $\delta > 0$, with probability at least $1 - \delta$, for every sample $\mathcal{S}$,

$$\left| \ell_{\exp}(h_{\mathcal{S}}) - \ell_{\emp}(h_{\mathcal{S}}) \right| \leq \epsilon(\mathcal{S}) + M \sqrt{\frac{2K \log(2) + 2 \log(1/\delta)}{|\mathcal{S}|}},$$

where $h_{\mathcal{S}}$, as before, denotes a function from the hypothesis class $\mathcal{H}$ returned by the learning algorithm given the data sample $\mathcal{S}$ of $\mathcal{Z}$. Additionally, M denotes a bound on the loss function ℓ.

Kawaguchi et al. [2022] improved the above generalization bound by establishing a data-dependent bound that reduces the dependency on K from $\sqrt{K}$ to $\log(K)$, as shown in [Kawaguchi et al., 2022, Theorem 1]. Even though ϵ-covers and partitions of a set are closely related notions, as each ϵ-cover trivially induces a partition, it may not yet be entirely clear how compactness can be leveraged to derive generalization bounds using the notion of robustness. This will become clearer later with Lemma 24.

The key insight in the following results is using more fine-grained expressivity properties of MPNNs compared to the 1-WL distinguishability that the VC dimension-based analyses used; see Section 3. Specifically, while Morris et al. [2023] leveraged the property that MPNN outputs for two graphs are identical whenever these graphs are 1-WL-equivalent, as shown in [Morris et al., 2019a], covering number approaches exploit a more "fine-grained" property of MPNNs. Namely, the MPNN outputs for two graphs are close in Euclidean distance whenever the graphs are close regarding some appropriately defined distance pseudo-metric between graphs.

Recipe for covering number-based MPNN generalization bounds Theorems 18 and 20 provide generalization bounds for any hypothesis space of Lipschitz continuous functions f, mapping data from a compact metric space $(\mathcal{M}, d_{\mathcal{M}})$ to $\mathbb{R}^d$, given an i.i.d. sample from any Borel distribution μ on $(\mathcal{M}, d_{\mathcal{M}})$. In the case of MPNNs and graph learning, the function f corresponds to the composition of the loss function ℓ with the MPNN's output h (after the readout layer).

There are two approaches in the literature for applying the theorems above in graph learning tasks. The first approach involves directly endowing the space of graphs $\mathcal{G}$ with a pseudometric
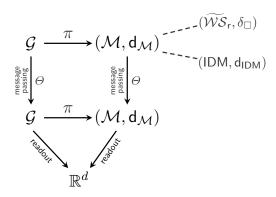
Figure 1: Commutative diagram illustrating the graph compact embedding process. In this diagram, the message-passing phase is represented as a function that updates node features, producing a graph with the same structure but updated node features.

$d_{\mathcal{G}}$ such that 1) MPNNs are Lipschitz continuous and 2) the completion of the space $\mathcal{G}$ under $d_{\mathcal{G}}$ gives a compact space. The fact that compact spaces always have a finite covering number leads to a generalization bound. Since the graphs' order is not bounded, we call this the *uniform regime*. The following technique for the uniform regime, which we call *the GNN Lipschitz-compact covering bounds*, was first introduced in Levie [2023]. The commutative diagram Figure 1 illustrates the following steps.

1. Define a compact space $(\mathcal{M}, d_{\mathcal{M}})$.

2. Define a mapping $\pi : \mathcal{G} \to \mathcal{M}$ that associates each graph $G \in \mathcal{G}$ with an element $\pi(G) \in \mathcal{M}$.

3. Consider the *pullback* pseudometric $\pi^* d_{\mathcal{M}}$ in $\mathcal{G}$ defined by

$$\pi^* d_{\mathcal{M}}(G_1, G_2) = d_{\mathcal{M}}(\pi(G_1), \pi(G_2)).$$

   Note that $\mathcal{G}$ is precompact with respect to $\pi^*(d_{\mathcal{M}})$, namely, the completion of $\mathcal{G}$ is compact.

4. For every $\text{MPNN}_{\Theta} : \mathcal{G} \to \mathbb{R}$ with parameters Θ, define a corresponding Lipschitz continuous mapping $\overline{\text{MPNN}}_{\Theta} : \mathcal{M} \to \mathbb{R}$, also called an MPNN, satisfying the equivariance relation
$$\pi(\text{MPNN}_{\Theta}(G)) = \overline{\text{MPNN}}_{\Theta}(\pi(G)).$$

   Note that the above construction assures that MPNN_{Θ} are Lipschitz continuous with respect to $\pi^*(d_{\mathcal{M}})$.

Then, since precompact spaces have finite covering (the same covering number as their compact completions), an upper bound on $\mathcal{N}(\mathcal{G}, \pi^* d_{\mathcal{M}}, \epsilon)$ is guaranteed, leading to a generalization bound through Theorem 18. This approach was introduced by Levie [2023], where $\mathcal{M}$ was the space of graphon-signals and $d_{\mathcal{M}}$ was the cut distance. Rauchwerger et al. [2024] later utilized this approach, where $\mathcal{M}$ was defined as the space of iterated degree measures (IDMs) and $d_{\mathcal{M}}$ the DIDM-mover's distance.

We note that making the space of graphs compact and MPNNs Lipschitz continuous can be challenging. Compactness requires "all graphs to be close to each other," and Lipschitz

continuity requires the graphs to be "sufficiently far apart for MPNNs to have bounded slopes." To avoid this complication, the second approach for GNN covering number generalization bounds involves restricting the focus to a finite space of graphs (e.g., $\mathcal{G}_n$), where finiteness immediately implies compactness for any pseudo-metric. In addition, various combinatorial techniques, such as the set-covering problem, can be employed on finite spaces to derive tight upper bounds or even compute the covering number exactly. This methodology is followed by Vasileiou et al. [2024], and since their bounds apply only to n-order graphs and not uniformly across the space of all graphs, we refer to these bounds as the *non-uniform regime*.

MPNNs hypothesis class The following works consider MPNNs with the following architecture. Each layer is defined using *order-normalized sum aggregation*, and we further use a mean readout layer, where

$$\boldsymbol{h}_v^{(t)} \coloneqq \varphi_t\Big(\boldsymbol{h}_u^{(t-1)}, \tfrac{1}{|V(G)|} \sum_{u \in N(v)} \boldsymbol{h}_u^{(t-1)}\Big) \quad \text{and} \quad \boldsymbol{h}_G \coloneqq \psi\Big(\tfrac{1}{|V(G)|} \sum_{u \in V(G)} \boldsymbol{h}_u^{(L)}\Big), \tag{4}$$

for $v \in V(G)$, where $\varphi_t \colon \mathbb{R}^{d_{t-1}+1} \to \mathbb{R}^{d_t}$ is an L_{φ_t}-Lipschitz continuous function with respect to the 2-norm-induced metric, for $d_t \in \mathbb{N}$ and $t \in [L]$. Additionally, $\psi \colon \mathbb{R}^{d_L} \to \mathbb{R}^d$ is an L_ψ-Lipschitz continuous function.[4] For some graph space $\mathcal{X}$ they analyze the class of L-layer MPNNs, based on Equation (4), where ψ is represented by a feedforward neural network (FNN) with Lipschitz constant L_{FNN} and bounded by $M' \in \mathbb{R}$, denoted as $\mathsf{MPNN}_{L,M',L_{\mathsf{FNN}}}^{\mathrm{ord}}(\mathcal{X})$:

$$\mathsf{MPNN}_{L,M',L_{\mathsf{FNN}}}^{\mathrm{ord}}(\mathcal{X}) \coloneqq \Big\{ h \colon \mathcal{X} \to \mathbb{R} \,\Big|\, h(G) \coloneqq \mathsf{FNN}_{\boldsymbol{\theta}} \circ \boldsymbol{h}_G, \text{ where } \boldsymbol{\theta} \in \Theta \Big\}.$$

Additionally, in the case of unattributed graphs, they simplify the architecture to the following:

$$\boldsymbol{h}_v^{(t)} \coloneqq \varphi_t\Big(1/|V(G)| \sum_{u \in N(v)} \boldsymbol{h}_u^{(t-1)}\Big),$$

which remains 1-WL-equivalent in distinguishing non-isomorphic graphs.

7.1. The uniform regime

Below, we present two works deriving generalization bounds via the uniform regime, as illustrated in Figure 1. We provide a detailed discussion of the work of Levie [2023], where the space $\mathcal{M}$ is chosen as the space of graphon-signals. Additionally, we briefly mention the approach taken by Rauchwerger et al. [2024], where $\mathcal{M}$ is the space of iterated degree measures.

Graphon-signals A graphon can be viewed as a generalization of a graph, where instead of discrete sets of nodes and edges, there is an infinite set indexed by the sets $V(W) \coloneqq [0, 1]$ for nodes and $E(W) \coloneqq V(W)^2 = [0, 1]^2$ for edges. A graphon is defined as a measurable symmetric function $W \coloneqq E(W) \to [0, 1]$, i.e., $W(x, y) = W(y, x)$. Each value $W(x, y)$ represents the probability or intensity of a connection between points x and y. Graphons are used to study the limit behavior of large graphs [Lovász, 2012]. A graphon-signal, introduced by Levie [2023], is a pair $(W, \boldsymbol{f})$, where W is a graphon and $\boldsymbol{f} \colon V(W) \to \mathbb{R}^d$ is a measurable function. We denote by

---

[4]As implied by Morris et al. [2019a], Xu et al. [2021], choosing the function φ_t appropriately leads to a 1-WL-equivalent MPNN layer.

$\mathcal{WS}^d$ the space of all graphon-signals with signals $\boldsymbol{f} \colon V(W) \to \mathbb{R}^d$. Additionally, for $r > 0$, we define the space of graphon-signals with r-bounded signal functions as $\mathcal{WS}_r^d = \mathcal{W} \times \mathcal{L}_r^\infty[0,1]$, where

$$\mathcal{L}_r^\infty[0,1] := \{\boldsymbol{f} \colon [0,1] \to \mathbb{R}^d \mid \|\boldsymbol{f}(x)\|_\infty \le r\}.$$

Any labeled graph can be identified with a corresponding graphon-signal as follows. Let (G, ℓ_G) be a labeled graph with node set $\{1, \dots, n\}$ and adjacency matrix $\boldsymbol{A}(G) = \{a_{i,j}\}_{i,j \in [n]}$. Let $\{I_k\}_{k=1}^N$ with $I_k = \left[\frac{k-1}{n}, \frac{k}{n}\right)$ be the equipartition of $[0,1]$ into n intervals. The graphon-signal $(W, f)_{(G, \ell_G)} = (W_G, f_{\ell_G})$ induced by (G, ℓ_G) is defined by

$$W_G(x,y) := \sum_{i,j=1}^n a_{ij} \, \mathbf{1}_{I_i}(x) \, \mathbf{1}_{I_j}(y) \quad \text{and} \quad f_{\ell_G}(z) := \sum_{i=1}^n \ell_G(i) \, \mathbf{1}_{I_i}(z),$$

where $\mathbf{1}_{I_i}$ is the indicator function of the set $I_i \subset [0,1]$. We write $(W, \boldsymbol{f})_{(G, \ell_G)} = (W_G, \boldsymbol{f}_{\ell_G})$ and identify any labeled graph with its induced graphon-signal.

Cut metrics. For any graphon W, we define the cut norm of W as

$$\|W\|_\square := \sup_{U,V \subset [0,1]} \left| \int_{U \times V} W(x,y) \, dx \, dy \right|,$$

where the supremum is over measurable sets $U, V \subset [0,1]$. The cut metric between two graphons $W, W' \in \mathcal{W}$ is defined as

$$d_\square(W, W') := \|W - W'\|_\square.$$

This measures the difference between the average edge weights of W and W' on the cut $U \times V$ that maximizes this difference. The *signal cut norm* on $\mathcal{L}_r^\infty[0,1]$ is defined by

$$\|f\|_\square := \sup_{S \subseteq [0,1]} \left| \int_S f(x) \, d\mu(x) \right|,$$

where the supremum is taken over the measurable subsets $S \subset [0,1]$. For simplicity of notation, we assume that signals map to $\mathbb{R}$ (i.e., $d = 1$), and we omit d from the notation. However, all results hold also for d-dimensional signals. The *graphon-signal cut norm* on $\mathcal{WS}_r$ is defined by

$$\|(W, f)\|_\square := \|W\|_\square + \|f\|_\square.$$

The *graphon-signal cut metric* and *graphon-signal cut distance* are defined by

$$d_\square((W, f), (V, g)) := \|(W, f) - (V, g)\|_\square,$$

and

$$\delta_\square((W, f), (V, g)) := \inf_{\psi \in S_{[0,1]}} d_\square((W, f), (V, g)^\psi),$$

where $(V, g)^\psi := (V^\psi, g^\psi)$ and $V^\psi(x,y) := V(\psi(x), \psi(y))$ and $V^\psi(x) := g(\psi(y))$, where $S_{[0,1]}$ is the set of all Lebesgue measure-preserving bijections on $[0,1]$ up to null sets.

Compactness and covering number of the graphon-signal space At this point, note that $(\mathcal{WS}_r, \delta_\square)$ is a pseudo-metric space. We denote the induced metric space on the space of $\delta_\square$-equivalence classes as $(\widetilde{\mathcal{WS}_r}, \delta_\square)$. Levie [2023], using a variant of the weak regularity lemma by Frieze and Kannan [1999], showed that the metric space $(\widetilde{\mathcal{WS}_r}, \delta_\square)$ is compact and derived an upper bound on its covering number.

Theorem 21 ([Levie, 2023, Theorem 3.6]). Let $r > 0$. The metric space $(\widetilde{\mathcal{WS}_r}, \delta_\square)$ is compact. Moreover, for every $c > 1$ and every sufficiently small $\epsilon > 0$, the space $\widetilde{\mathcal{WS}_r}$ can be covered by

$$\kappa(\epsilon) \le 2^{k^2}$$

balls of radius ϵ, where $\kappa(\epsilon) := \mathcal{N}(\widetilde{\mathcal{WS}_r}, \delta_\square, \epsilon)$ and $k := \lceil 2^{2c/\epsilon^2} \rceil$.

Lipschitz continuity of MPNNs on graphon-signals To leverage the compactness result for analyzing MPNNs, Levie [2023] extended the definition of MPNNs to graphon-signals so that MPNNs on attributed graphs become a special case. In the context of the diagram in Figure 1, $\pi(G, \ell_G) = (W_G, \boldsymbol{f}_{\ell_G})$ is the operator that induces graphon-signals from a graph signal or every MPNN without readout h, and for any labeled graph (G, ℓ_G), we have $\pi(h_{(G,\ell_G)}) = h_{\pi(G,\ell_G)}$. As a result, for MPNNs with readout $h \in \mathsf{MPNN}^{\mathrm{ord}}_{L,M',L_{\mathsf{FNN}}}(\mathcal{X})$,[5]

$$h_{(G,\ell_G)} = h_{(W_G, \boldsymbol{f}_{\ell_G})}.$$

See Appendix D for a detailed description of MPNNs applied to graphon-signals and an extended analysis of graphon theory.

One of the main results of Levie [2023] is that MPNNs defined on graphon-signals are Lipschitz continuous concerning the graphon-signal cut distance. Furthermore, the Lipschitz constant of an MPNN can be bounded by the Lipschitz constants of its message functions. It is important to note that in the extension from graphs to graphons, the aggregation step in MPNNs must always be normalized by order of the graph (i.e., $\mathsf{MPNN}^{\mathrm{ord}}_{L,M',L_{\mathsf{FNN}}}(\mathcal{X})$); otherwise, the commutative diagram (Figure 1) cannot be satisfied.

Generalization bound for MPNNs As a consequence of the Lipschitz continuity of MPNNs, the compactness of the space of graphon-signals, and the bound on its covering number, the following generalization theorem is established for the binary classification of attributed graphs.

Theorem 22 (Levie [2023, Theorem 4.2]). Consider the hypothesis class $\mathcal{H}$ of Lipschitz continuous MPNNs with Lipschitz constant L_1. Suppose the loss function is also Lipschitz continuous with Lipschitz constant L_2. Let $X_1, \ldots, X_N$ be independent random samples from a distribution μ on $\widetilde{\mathcal{WS}_r}$. Then, for every $\delta > 0$, with probability at least $1 - \delta$, the following holds: for all $h \in \mathcal{H}$

$$\left| \ell_{\mathrm{emp}}(h) - \ell_{\exp}(h) \right| \le \xi^{-1}(N/4)\Big(2L + \frac{1}{\sqrt{2}}(L + M)\big(1 + \sqrt{\log(2/\delta)}\big)\Big), \qquad (5)$$

where $L := L_1 L_2$, M is an upper bound for the loss function, $\xi(\epsilon) := \frac{\kappa(\epsilon)^2 \log(\kappa(\epsilon))}{\epsilon^2}$, $\kappa(\epsilon)$ is an upper bound on the covering number $\mathcal{N}(\widetilde{\mathcal{WS}_r}, \delta_\square, \epsilon)$, and ξ^{-1} is the inverse function of ξ.

---

[5]To be precise, Levie [2023] defined MPNNs on graphons through a slightly more general family of MPNNs.

Note that the term $\xi^{-1}(N/2C)$ in Equation (5) decreases to zero as the size of the training set N goes to infinity, but very slowly. The advantage of this theorem is its generality: it does not depend on any assumption on the data distribution. For example, the data distribution may be any arbitrary probability distribution supported on the subset of attributed graphs. This makes Theorem 22 a general uniform generalization bound for MPNNs on attributed graphs. The disadvantage is that in such a general setting, the decay concerning the size of the training set N is very slow.

Comparison to iterated degree measures approach Rauchwerger et al. [2024] followed a similar approach by mapping the space of graphs into the space of distributions of iterated degree measures (DIDMs). They defined a metric, called the DIDM-mover's distance, that renders this space compact and introduced an MPNN framework on DIDMs in which MPNNs are Lipschitz continuous while satisfying the commutative diagram (Figure 1). We omit the details since this construction parallels the graphon-based approach and involves slightly complicated notation. However, we briefly compare the results of these two works. Levie [2023] derived an explicit bound on the covering number using the weak regularity lemma, whereas Rauchwerger et al. [2024] only established the finiteness of the covering number through compactness. On the other hand, the graphon-signal cut distance underlies a finer topology than the metric used in Rauchwerger et al. [2024],[6] which may potentially mean that the covering number w.r.t. the DIDM-mover's distance is smaller than that of cut distance.

7.2. The non-uniform regime

Below, we present an approach for deriving generalization bounds for MPNNs by restricting our focus to the space of n-order graphs and leveraging the notion of robustness from Theorem 20. We begin by demonstrating how Theorem 20 can be transformed into a generalization bound that depends on the covering number of a finite space and the corresponding radius. We then present the covering number bounds derived in this framework, followed by the final form of the generalization bounds for MPNNs. For simplicity, we show only the case of graphs without node features, where the corresponding pseudometrics are more straightforward to define.

Unlabeled Graph Metrics Following the works of Böker [2021], Böker et al. [2023], we define the *tree distance* as

$$\delta_{\|\cdot\|}^{\mathcal{T}}(G, H) \coloneqq \min_{\boldsymbol{S} \in D_n} \|\boldsymbol{A}(G)\boldsymbol{S} - \boldsymbol{S}\boldsymbol{A}(H)\|_\square,$$

where D_n denotes the set of $n \times n$ doubly stochastic matrices (i.e., matrices with non-negative entries where each row and each column sums to 1), and the cut-norm on $\mathbb{R}^{n \times n}$ is defined as

$$\|\boldsymbol{M}\|_\square \coloneqq \max_{S \subset [n], T \subset [n]} \Big| \sum_{i \in S, j \in T} m_{ij} \Big|.$$

Continuity of MPNNs on graphs A slight difference with previous works is that in Vasileiou et al. [2024], the MPNNs are required to satisfy a weaker notion of continuity than Lipschitz continuity, namely equicontinuity. The following result by Vasileiou et al. [2024] shows that

---

[6]Every open set w.r.t. DIDM-mover's distance is also open w.r.t. cut distance, but not vice-versa. Hence, every open covering w.r.t. DIDM-mover's distance is also a covering w.r.t. cut distance, but not vice-versa.

the MPNN class $\mathsf{MPNN}^{\mathrm{ord}}_{L,M',L_{\mathsf{FNN}}}(\mathcal{G}_n)$ satisfies the equicontinuity property concerning the tree distance.

Lemma 23. For all $\varepsilon > 0$, $L \in \mathbb{N}$, there exists a function $\gamma(\varepsilon) > 0$ such that for $n \in \mathbb{N}$, $M' \in \mathbb{R}$, and $G, H \in \mathcal{G}_n$, and for all MPNN architectures in $\mathsf{MPNN}^{\mathrm{ord}}_{L,M',L_{\mathsf{FNN}}}(\mathcal{G}_n)$, the following implication holds,

$$\delta^{\mathcal{T}}_{\square}(G, H) < n^2 \cdot \gamma(\varepsilon) \implies \|\boldsymbol{h}_G - \boldsymbol{h}_H\|_2 \leq \varepsilon.$$

Based on the above result, they define the following non-decreasing function $\bar{\gamma} \colon \mathbb{R}^+ \cup \{+\infty\} \to \mathbb{R}^+ \cup \{+\infty\}$ as $\gamma(+\infty) = +\infty$, and

$$\bar{\gamma}(\epsilon) \coloneqq \sup\Big\{\delta > 0 \mid \delta^{\mathcal{T}}_{\square}(G, H) \leq n^2\delta \implies \|h_G - h_H\| \leq \epsilon, h \in \mathsf{MPNN}^{\mathrm{ord}}_{L,M',L_{\mathsf{FNN}}}(\mathcal{G}_n), \; \forall G, H \in \mathcal{G}_n\Big\},$$

and define its generalized inverse function $\bar{\gamma}^{\leftarrow} \colon \mathbb{R}^+ \cup \{+\infty\} \to \mathbb{R}^+ \cup \{+\infty\}$, where $\mathbb{R}^+ = (0, +\infty)$, as

$$\bar{\gamma}^{\leftarrow}(y) \coloneqq \inf\{\epsilon > 0 \mid \gamma(\epsilon) \geq y\}.$$

Finally, by establishing that robustness is guaranteed by the equicontinuity property, they derive the following generalization bounds for the class $\mathsf{MPNN}^{\mathrm{ord}}_{L,M',L_{\mathsf{FNN}}}(\mathcal{G}_n)$, showing the connection between the covering number and the parameters in the definition of robustness in the binary classification setting using the binary cross entropy loss with the sigmoid function. That is $\ell \colon \mathbb{R} \times \{0, 1\} \to \mathbb{R}$, $\ell(x, y) = y \log(\sigma(x)) + (1 - y) \log(1 - \sigma(x))$.

Lemma 24. For $\varepsilon > 0$, $n, L \in \mathbb{N}$, and $M' \in \mathbb{R}$, any graph learning algorithm for the class $\mathsf{MPNN}^{\mathrm{ord}}_{L,M',L_{\mathsf{FNN}}}(\mathcal{G}_n)$ is

$$\Big(2\mathcal{N}(\mathcal{G}_n, \delta^{\mathcal{T}}_{\square}, \varepsilon), L_\ell \cdot L_{\mathsf{FNN}} \cdot \bar{\gamma}^{\leftarrow}\Big(\frac{2\varepsilon}{n^2}\Big)\Big)\text{-robust.}$$

Hence, for any sample $\mathcal{S}$ and $\delta \in (0, 1)$, with probability at least $1 - \delta$, we have

$$|\ell_{\exp}(h_{\mathcal{S}}) - \ell_{\mathrm{emp}}(h_{\mathcal{S}})| \leq L_\ell \cdot L_{\mathsf{FNN}} \cdot \bar{\gamma}^{\leftarrow}\Big(\frac{2\varepsilon}{n^2}\Big) + M\sqrt{\frac{4\mathcal{N}(\mathcal{G}_n, \delta^{\mathcal{T}}_{\square}, \varepsilon)\log(2) + 2\log\big(\frac{1}{\delta}\big)}{|\mathcal{S}|}},$$

where M is an upper bound for the loss function ℓ and L_ℓ is the Lipschitz constant of $\ell(\cdot, y), y \in \{0, 1\}$.

Generalization bound Since computing the covering number as a function of ε is challenging due to the complex structure of this pseudo-metric space, Vasileiou et al. [2024] bounded the covering number by a function of ε and m_n (i.e., the number of 1-WL-equivalence classes in $\mathcal{G}_n$) for a given n. This leads to a generalization bound that depends solely on ε, enabling the computation of the optimal radius by solving a minimization problem. Specifically, for the set of n-order graphs $\mathcal{G}_n$ and a radius of $\varepsilon = 4k$, they constructed a cover of $\mathcal{G}_n$ with cardinality $\frac{m_n}{k+1}$, for all $k \in \mathbb{N}$, yielding the following family of generalization bounds.

Theorem 25. For $n, L \in \mathbb{N}$, $M' \in \mathbb{R}$, and any graph learning algorithm for $\mathsf{MPNN}^{\mathrm{ord}}_{L,M',L_{\mathsf{FNN}}}(\mathcal{G}_n)$, and for any sample $\mathcal{S}$ and $\delta \in (0, 1)$, with probability at least $1 - \delta$, we have

$$|\ell_{\exp}(h_{\mathcal{S}}) - \ell_{\mathrm{emp}}(h_{\mathcal{S}})| \leq L_\ell \cdot L_{\mathsf{FNN}} \cdot \bar{\gamma}^{\leftarrow}\Big(\frac{8k}{n^2}\Big) + M\sqrt{\frac{4\frac{m_n}{k+1}\log(2) + 2\log\big(\frac{1}{\delta}\big)}{|\mathcal{S}|}},$$

for $k \in \mathbb{N}$, where M is an upper bound on the loss function ℓ.

The above bound shows a linear decrease in the covering number as a function of ε. Furthermore, it is demonstrated that restricting the graph family makes it possible to achieve a bound that decreases exponentially with the radius. Additionally, they identify graph families for which the bounds can achieve a non-constant improvement compared to those in Morris et al. [2023] (i.e., $\frac{m_n}{n}$ instead of m_n).

Vertex-attributed graphs and other Aggregation Functions   To account for vertex-attributed graphs and consider the number of layers in an MPNN architecture, Vasileiou et al. [2024] define the *forest distance*. They show that this distance is an alternative but equivalent variation of the Tree Mover's distance introduced in Chuang and Jegelka [2022], allowing them to devise an analogous version of Lemma 24 for vertex-attributed graphs, taking layer information into account. Additionally, they derive a variant of 1-WL that precisely captures the computation of the MPNN layers using mean aggregation, i.e., dividing by the number of neighbors after summing over the neighboring features, and they extend their generalization bounds to mean aggregation MPNNs.

Overall, the covering number bounds in this work are significantly tighter than those in Levie [2023], which is expected due to the considerably more restricted space of graphs considered here. In practice, they experimentally show that these bounds closely approximate the actual test error observed on different datasets. These results also account for mean-aggregation MPNNs, which were not analyzed in the other works.

7.3. Generalization bounds on mixtures of generating graphons

The uniform convergence rate of Theorem 18 is very slow under no assumption on the data distribution. To improve the asymptotics and obtain a bound, which can be meaningful in some practical settings, Maskey et al. [2022, 2024] introduces assumptions on the data distribution.

Graphons as generative graph models   A graphon-signal can serve as a generative model of graphs. Given a graphon-signal $(W, \boldsymbol{f})$, a random graph with n nodes is generated by sampling uniformly and independently the points $\{u_i\}_{i=1}^n$ from $[0, 1]$, and connecting each pair u_i, u_j with probability $W(u_i, u_j)$ to obtain the edges of the graph. The node feature of node u_i is $\boldsymbol{f}(u_i)$, for $i \in [n]$.

Maskey et al. [2022] assumed that there is a finite set of template graphon-signals $(W_j, f_j)_{j=1}^{\Gamma}$, such that each graph from the data distribution is sampled independently by choosing randomly a template j, a degree n (from a distribution of degrees ν), and sampling a graph of n nodes from (W_j, f_j). In fact, by the Aldous-Hoover theorem [Aldous, 1981, 1985, Hoover, 1979, 1982], any data distribution on graphs can be modeled this way if one allows a general "infinite" distribution of graphon-signal templates instead of a finite set.

In a classification setting, it is assumed that graphs sampled from the same template belong to the same class but not necessarily vice versa. Maskey et al. [2024] extends the analysis by considering sparse graphs sampled with noise from the template graphon-signals. This generative model of the data is called *mixture of graphons*. A generalization theorem can then be derived from this setting. Let μ^N be the probability space of randomly sampled training sets of N attributed graphs, $\mathrm{MPNN}_{L_\varphi}$ the hypothesis class of MPNN with Lipschitz message passing functions with Lipschitz constant L_φ, ν the probability measure of the choice of the number of nodes n of the graphs, ℓ the loss function, bounded by $M > 0$ with Lipschitz constant C_ℓ, ε the

noise rate when generating graphs, and α the sparsity level of the sampled graphs, where $n^{1-\alpha}$ is proportional to the average node degree; see [Maskey et al., 2024] for more details.

Theorem 26 ([Maskey et al., 2024] Generalization of MPNN on mixtures of graphons)**.** There exist constants $C, C' > 0$ that describe the complexity of the hypothesis class of MPNNs in terms of their depth and the Lipschitz constants of their message functions, such that

$$
\begin{aligned}
\mathbb{E}_{\mu^m} & \left[\sup_{f \in \mathrm{MPNN}_{L_\varphi}} \left(\ell_{\mathrm{emp}}(f) - \ell_{\mathrm{exp}}(f) \right)^2 \right] \leq \frac{2^\Gamma M^2 \pi}{N} \\
& + \frac{2^\Gamma C_\ell^2}{N} \left(C \cdot \mathbb{E}_{n \sim \nu} \left[\frac{1 + \log(n)}{n} n^{2\alpha} + \frac{1 + \log(n)}{n^{1/(D_\chi + 1)}} n^{2\alpha} + \mathcal{O}(\exp(-n)) \right] + C' \varepsilon \right),
\end{aligned}
\tag{6}
$$

where D_χ is a constant that characterizes the (Minkowski) dimension of the geometries described by the template graphons.

The proof of Theorem 26 is based on a covering number construction, a special case of Theorem 20 first proven in [Maskey et al., 2024, Appendix C.2]. Note that the first term in the right-hand side of Equation (6) is typically much smaller than the second term, as it does not depend on the complexity C of the hypothesis class. Hence, one insight that can be derived from Equation (6) is that MPNNs generalize better the larger the graphs in the data distribution are. Namely, even if the complexity of the MPNNs C is higher than the sample size m, one can still generalize well if the average number of nodes n is high. The disadvantage of this bound is that it is specific for a mixture of a graphon model, which may not capture the behavior of some real-life data distributions if the number of classes K is small.

8. Transduction in graph learning

Up to this point, our analysis of the generalization performance of MPNNs has primarily focused on graph-level prediction tasks. However, node-level prediction problems are equally important in real-world applications, such as social networks, recommendation systems, and more [Wu et al., 2023a]. Scarselli et al. [2018], Verma and Zhang [2019], Garg et al. [2020], Zhou and Wang [2021] describe how their generalization bounds can be extended to node classification tasks by converting graphs into subgraphs via local node-wise computation trees and treating them as i.i.d. samples. This assumption does not align with the commonly seen graph-based semi-supervised learning setting. Therefore, more recent research has derived generalization bounds for MPNNs in node classification under a *transductive learning approach*. In the first (to our knowledge) work on a transductive learning approach to node classification [Oono and Suzuki, 2020], they provide generalization bounds via the transductive Rademacher complexity for *multi-scale MPNNs*. Multi-scale MPNNs are MPNN architectures designed to mitigate the over-smoothing issue commonly observed in MPNNs. Similarly, Cong et al. [2021] established generalization bounds for GNNs by leveraging transductive uniform stability under the full-batch gradient descent. Further, Esser et al. [2021] analyzed the generalization properties of graph convolutional networks via transductive Rademacher complexity, highlighting the interplay of graph structure and the predictive performance of MPNNs. Tang and Liu [2023] also developed generalization bounds in the transductive setting, applicable to models trained with the SGD. Before we define the statistical framework of transductive learning and present some well-established generalization bounds, let us briefly highlight the core distinction between

transductive and inductive learning. Inductive learning, described so far, aims to learn a function h from a training dataset and then applies h to new, unseen data, emphasizing the generalization beyond the training examples. In contrast, transductive learning is tailored to make predictions specifically for a fixed set of unseen data available during training, focusing on optimizing predictions only for this set.

In the following, we present the transductive learning framework for graphs, defining the transductive Rademacher complexity as a modification of the traditional Rademacher complexity introduced in Section 4 and explaining the key motivations behind this extension. We also present the complexity bounds from Esser et al. [2021] as a specific example. We choose to highlight this work due to the importance of the transductive Rademacher complexity, a fundamental concept in transductive learning, as described in [Koltchinskii, 2001, El-Yaniv and Pechyony, 2007].

Transductive learning statistical framework   Let $G \in \mathcal{G}_{n,d}^{\mathbb{R}}$. For ease of notation, we assume $V(G) = [n]$, and we define the matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, with rows $\boldsymbol{x}_i$ representing the feature vectors of nodes $i \in [n]$. Without loss of generality, for some $m \in [n]$, we set $u = n - m$, we assume that the labels $y_1, \ldots, y_m \in \{-1, +1\}$ are known, and the goal is to predict $y_{m+1}, \ldots, y_n$. Usually, in the transductive framework, we do not assume that the data is sampled from an unknown underlying distribution. Instead, we introduce randomness in the data generation process as follows. We assume that the labeled set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ is randomly sampled from all possible m-size subsets of $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$. Or equivalently, we may assume that a random permutation of $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ is chosen, and the first m pairs are selected as the training set, while the remaining pairs form the test set. As a result, the training data is no longer independently sampled, and the underlying distribution is known. Note that the learner still has access to the feature vectors $\{\boldsymbol{x}_i\}_{i=m+1}^n$. See [Vapnik, 2006, Section 10.1], [Pechyony, 2008], for different mechanisms of training set generation and different learning goals. For a given loss function ℓ defined as previously, we define the test error of a predictor h in some hypothesis class $\mathcal{H}$ as $\ell_u(h) := \frac{1}{n-m} \sum_{i=m+1}^n \ell(h, \boldsymbol{x}_i, y_i)$ and the empirical error as $\ell_{\mathrm{emp}}(h) := \frac{1}{m} \sum_{i=1}^m \ell(h, \boldsymbol{x}_i, y_i)$. The goal is then to bound their difference, i.e., deriving bounds of the form

$$\ell_u(h) \leq \ell_{\mathrm{emp}}(h) + \mathrm{complexity}(\mathcal{H}, G).$$

Comparison to inductive learning   Before proceeding to the transductive Rademacher complexity bounds derived by Esser et al. [2021], we provide some remarks on key similarities and differences between transductive and inductive learning. First, note that the data is assumed to be sampled independently in the inductive scenario, which is not valid in the transductive scenario. In both settings, the training points are sampled from some distribution. However, in the inductive case, this unknown distribution is defined over the arbitrary product space $\mathcal{X} \times \mathcal{Y}$. In contrast, in the transductive setting, the distribution is known and is defined over a finite subset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$. In transductive learning, we also evaluate the error between the empirical loss and the loss over a fixed test set, denoted by ℓ_u, which strongly depends on the training sample. In contrast, in the inductive setting, the error is measured between the empirical loss and the expected loss, which is an unknown but constant quantity. Thus, in the transductive setting, we aim to bound the difference between two random variables, i.e., $\ell_{\mathrm{emp}} - \ell_u$, while in the inductive setting, the objective is to bound the difference between a random variable and its expectation, i.e., $\ell_{\mathrm{emp}} - \ell_{\mathrm{exp}}$.

Transductive Rademacher complexity bounds Assume the binary classification node level setting and, with a slight abuse of notation, that the hypothesis class consists of functions from $\boldsymbol{X}$ to $\{-1, 1\}^n$. Furthermore, since typically neural networks output a soft predictor in $\mathbb{R}$ transformed to labels via a sign function, we focus on predictors $h \colon \mathbb{R}^{n \times d} \to \mathbb{R}^n$. Following Esser et al. [2021], for some $L \in \mathbb{N}$, we consider the class of L-layer MPNNs with the following architecture. In each layer t, we compute the updated node features $\boldsymbol{g}_t \in \mathbb{R}^{n \times d_t}$, where d_t is the dimension of layer t as follows. For $\boldsymbol{S} := \boldsymbol{A}(G) + \boldsymbol{I}_{n \times n}$, with $\boldsymbol{A}(G)$ being the adjacency matrix of the graph G, and ϕ denote point-wise activation function, which we assume to be a Lipschitz function with Lipschitz constant L_ϕ, we define the hypothesis class over all L-layer MPNNs as

$$\mathcal{H}_G^\phi := \left\{ h^\phi(\boldsymbol{X}) = \boldsymbol{g}_L \colon \mathbb{R}^{n \times d} \to \mathbb{R}^n \right\}$$

with

$$\boldsymbol{g}_t := \phi(\boldsymbol{b}_t + \boldsymbol{S}\boldsymbol{g}_{t-1}\boldsymbol{W}_t), \quad t \in [L], \quad \boldsymbol{g}_0 \equiv \boldsymbol{X},$$

where $\boldsymbol{W}_t \in \mathbb{R}^{d_{t-1} \times d_t}$ and $\boldsymbol{b}_t \in \mathbb{R}^n$ are the trainable weight matrix and bias term, respectively. Through the message-passing process, it becomes clear that the graph structure is treated as part of the hypothesis class through the adjacency matrix. By further assuming that the norms $\|\boldsymbol{b}_t\|_1$ and $\|\boldsymbol{W}_t\|_\infty$ of the trainable parameters are bounded by β and $\omega \in \mathbb{R}$, respectively, for all $t \in [L]$, we denote our hypothesis class with $\mathcal{H}_G^{\phi, \beta, \omega}$.

We now define the *transductive Rademacher complexity* and describe the differences between the inductive Rademacher complexity introduced in Section 4.

Definition 27 (TRC [El-Yaniv and Pechyony, 2007])**.** Let $\mathcal{V} \subset \mathbb{R}^n$, $p \in [0, 0.5]$, and m be the number of labeled points. Let $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)^\intercal$ be a vector of independent and identically distributed random variables, each taking the values $+1$ and -1 with probability p, and 0 with probability $1 - 2p$. The *transductive Rademacher complexity* (TRC) of $\mathcal{V}$ is defined as

$$\mathcal{R}_{m,n}(\mathcal{V}) := \left(\frac{1}{m} + \frac{1}{n-m} \right) \cdot \mathbb{E}_\sigma \left[\sup_{\boldsymbol{v} \in \mathcal{V}} \boldsymbol{\sigma}^T \boldsymbol{v} \right].$$

Some key differences from inductive Rademacher complexity include that transductive Rademacher complexity depends on both the training and the test points. In contrast, the inductive complexity only depends on training points. Additionally, the transductive complexity is an empirical quantity that does not depend on any underlying distribution, only on the known distribution of σ.

El-Yaniv and Pechyony [2007] established a generalization error bound through the Transductive Rademacher Complexity (TRC) that is similar to Theorem 7. Subsequently, Esser et al. [2021] bounded the TRC of $\mathcal{H}_G^{\phi, \beta, \omega}$ and, using this result, showed the following generalization bound:

Theorem 28 ([Esser et al., 2021])**.** For the transductive Rademacher complexity of the hypothesis class $\mathcal{H}_G^{\phi, \beta, \omega}$, we have the following bound,

$$\mathcal{R}_{m,n}(\mathcal{H}_G^{\phi, \beta, \omega}) \leq \frac{c_1 n^2}{m(n-m)} \left(\sum_{t=0}^{L-1} c_2^t \|\boldsymbol{S}\|_\infty^t \right) + c_3 c_2^L \|\boldsymbol{S}\|_\infty^L \|\boldsymbol{S}\boldsymbol{X}\|_{2 \to \infty} \sqrt{\log(n)},$$

where $c_1 := 2L_\phi \beta$, $c_2 := 2L_\phi \omega$, $c_3 := L_\phi \omega \sqrt{2/d}$, and $\| \cdot \|_{2 \to \infty}$ denotes the maximum 2-norm of any column. Following El-Yaniv and Pechyony [2007], the generalization error bound becomes,

for any $\delta > 0$, with probability at least $1 - \delta$,

$$\ell_{\mathrm{u}}(h) - \ell_{\mathrm{emp}}(h) \leq \mathcal{R}_{m,n}(\mathcal{H}_G^{\phi,\beta,\omega}) + c_4 \frac{n\sqrt{\min\{m, n-m\}}}{m(n-m)} + c_5 \sqrt{\frac{n}{m(n-m)} \ln\left(\frac{1}{\delta}\right)},$$

where c_4, c_5 are absolute constants such that $c_4 < 5.05$, and $c_5 < 0.8$.

The remainder of this work focuses on understanding how the graph structure influences the generalization performance by analyzing the dependence of the above bound on $\boldsymbol{S}$ and $\boldsymbol{X}$.

9. Various frameworks

So far, we have reviewed works analyzing the generalization abilities of MPNNs through common theoretical frameworks such as VC dimension, margin-based bounds, Rademacher complexity, PAC-Bayesian framework, stability-based analysis, graphon theory, and transductive learning. However, novel approaches beyond these established frameworks also yield significant generalization results.

In this section, we focus on research that investigates the generalization properties of GNNs, specifically using methods that diverge from the ones previously mentioned. For instance, Zhang et al. [2020] approaches generalization by assuming the existence of a ground truth MPNN model with zero generalization error. They propose a gradient-based optimizer to learn this model and guarantee MPNN generalizability. One limitation of this work is that it only considers models with a single hidden layer. Li et al. [2022a] introduce a different framework, combining SGD with graph topology sampling techniques. They provide generalization bounds for GCNs with up to three layers, primarily in the context of node classification tasks. Ma et al. [2021] employ a PAC-Bayesian approach in the semi-supervised setting, deriving generalization bounds for node classification under non-i.i.d. assumptions. Similarly, Lee et al. [2024] apply the PAC-Bayesian framework to derive generalization bounds for a wide range of knowledge graph representation learning methods [Bordes et al., 2013, Chung et al., 2023]. In a novel study, Suzuki et al. [2023] modify the Rademacher complexity to derive generalization bounds for GNNs using non-Euclidean metric spaces for graph embeddings. In another recent work, Shi et al. [2024] take a step beyond the standard statistical learning framework by utilizing random matrix theory to account for phenomena like the double descent, establishing generalization bounds for GNNs. Furthermore, Duranthon and Zdeborová [2024] analyze the generalization performance of single-layer GCNs in the high-dimensional setting, using data generated by stochastic block models. Li et al. [2024a] build on the margin-based generalization framework proposed by Chuang et al. [2021], which is based on k-Variance and the Wasserstein distance. They provide a method to analyze how expressiveness affects graph embeddings' inter- and intra-class concentration. Pellizzoni et al. [2024] investigate node-individualized MPNNs via the covering number framework [Xu and Mannor, 2012] outlined in Section 7. Kriege et al. [2018] leverage results from graph property testing [Goldreich, 2010] to study the sample complexity of learning to distinguish various graph properties, e.g., planarity or triangle freeness, using graph kernels [Borgwardt et al., 2020, Kriege et al., 2020]. Xu et al. [2020] formulate an early framework for learning algorithmic tasks with intermediate supervision. Finally, Keriven et al. [2020] investigate the generalization capabilities of GNNs on large random graphs.

10. Out-of-distribution generalization

So far, we have discussed the generalization analysis of GNNs in scenarios where both the training and test graphs are sampled from the same distribution, known as *in-distribution generalization*. However, in practice, it is often the case that the training graphs are drawn from a distribution that slightly differs from the distribution of the graphs where the GNN is expected to perform well. This challenge arises due to a distribution shift between the training and test sets, emphasizing the importance of studying *out-of-distribution* (OOD) generalization. Hence, in this section, we highlight key works that address the OOD generalization capabilities of GNNs. We first describe the statistical learning setting, emphasizing the differences between in- and out-of-distribution generalization problems.

Statistical learning setting Let $\mathcal{G}$ be a graph input space and $\mathcal{Y}$ be the label space. We focus on predictors $f_\theta \colon \mathcal{G} \to \mathcal{Y}$, with learnable parameters θ and a loss function ℓ. While we primarily consider graph-level tasks, it is straightforward that the input space could consist of nodes or edges for node-level or edge-level tasks, respectively. Given a training set $\mathcal{S} = \{(G_i, y_i)\}_{i=1}^N$ of N i.i.d. samples from a distribution P_{train} on $\mathcal{G} \times \mathcal{Y}$, the goal is to learn an optimal predictor f_θ^* that minimizes the expected loss on data drawn from a different test distribution P_{test}, where $P_{\text{train}} \neq P_{\text{test}}$, i.e.,

$$f_\theta^* := \arg\min_{f_\theta} \mathbb{E}_{P_{\text{test}}}[\ell(f_\theta(G), y)].$$

Since the assumption that the training and test data are i.i.d. no longer holds, traditional methods for bounding the generalization gap using concentration inequalities are not applicable. Consequently, less theoretical work has been conducted on out-of-distribution (OOD) generalization than in-distribution generalization. Even defining the generalization gap in this setting is unclear due to two distinct underlying distributions. Ye et al. [2021] developed a rigorous and quantitative framework for defining OOD generalization using tools from information theory. Li et al. [2022b] provide a comprehensive overview of various OOD generalization approaches in graph learning. By viewing graph data as a distribution over computation graphs and using optimal transport, Chuang and Jegelka [2022] quantified the effect of input perturbations through a Lipschitz constant and a domain generalization bound. Following a causal machine learning approach, Gui et al. [2023] analyzed the out-of-distribution generalization of GNNs. Other works analyzing OOD generalization include Xu et al. [2021], which provided sufficient conditions (on the architecture and task) under which GNNs generalize well, and Li et al. [2024b], which investigated data augmentation approaches.

A concept closely related to, but distinct from, out-of-distribution generalization is *size-generalization* or *size-transferabilitya*. Size-transferability analyzes the ability of GNNs to generalize across different graph sizes. It addresses the question: "When do GNNs generalize to graphs of sizes not seen during training?" Continuous extensions of graphs, such as graphons and graphops, appear to be appropriate tools for analyzing the size-transferability of GNNs. These continuous counterparts of graphs retain structural information without being affected by size variations. Some of the most detailed analyses of size-transferability include Levie et al. [2019], Ruiz et al. [2020], Levie et al. [2021], Le and Jegelka [2023]. Herbst and Jegelka [2025] analyze transferability of higher-order graph neural networks. Taking a different approach, Yehudai et al. [2021] studied graph distributions in which local structures (which GNNs capture) significantly depend on graph size. Specifically, they used d-patterns (a notion closely related to the k-neighborhood of a node) to show that a d-pattern discrepancy between the training

graphs and the test set can lead to poor global minima, ultimately failing to generalize to larger graphs. Furthermore, Bevilacqua et al. [2021] investigated size generalization using a causal machine learning approach, assuming independence between cause and mechanism, i.e., $P_{\text{train}}(Y|X) = P_{\text{test}}(Y|X)$, drawing on principles from Louizos et al. [2017].

11. Future directions and open problems

The present work shows that the last years resulted in many works studying MPNNs' generalization properties. However, many open problems and challenges remain. First, it is valuable to understand if tighter generalization can be obtained when restricting the analysis to specific graph classes, e.g., trees or planar graphs. As a first step, one could tailor the results of Morris et al. [2023] to specific graph classes. Moreover, most generalization bounds, e.g., based on VC dimension theory or Rademacher averages, only consider simplistic graph parameters such as the maximum degree. Hence, it is essential to include more structural information in the graph. In addition, only some works study the generalization properties of more expressive GNNs beyond 1-WL's limitations. While Franks et al. [2023] took a first step in that direction by studying simple, more expressive MPNNs, including subgraph information, the paper only studies the linearly separable case. Hence, it would also be valuable to understand under which conditions adding more expressive power to MPNNs leads to better generalization performance. Since most of the discussed bounds are based on VC dimension theory or a similar framework, they are vacuous, not implying much for practical considerations. Hence, it is essential to adapt learning-theoretic generalization bounds, such as information theoretic [Chu and Raginsky, 2023] or compression-based frameworks [Arora et al., 2018], to the graph domain and understand to what extent one can incorporate graph structure information in such bounds. We also refer the reader to Morris et al. [2024], which provides an insightful resource for open problems and challenges in the foundations of graph learning.

12. Conclusion

Here, we reviewed the current state of generalization theory for graph learning. Specifically, we overviewed generalization theory for MPNNs using the VC dimension, Rademacher complexity, PAC-Bayesian theory, and covering numbers. Moreover, we reviewed generalization bounds based on the stability of MPNNs on graphs. In addition, we surveyed recent work on out-of-distribution generalization. Finally, we discussed open problems and challenges to advance our understanding of MPNNs' generalization. We hope that this paper presents a valuable handbook for the generalization theory of MPNNs and helps advance our understanding further.

Acknowledgements

References

D. J. Aldous. Representations for partially exchangeable arrays of random variables. *Journal of Multivariate Analysis*, 11(4):581–598, 1981.

D. J. Aldous. Exchangeability and related topics. In *École d'Été de Probabilités de Saint-Flour XIII — 1983*, pages 1–198, 1985.

N. Alon, S. Hanneke, R. Holzman, and S. Moran. A theory of PAC learnability of partial concept classes. In *FOCS*, 2021.

M. Anthony and P. L. Bartlett. *Neural Network Learning - Theoretical Foundations*. Cambridge University Press, 2002.

S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. Stronger generalization bounds for deep nets via a compression approach. In *ICML*, 2018.

V. Arvind, J. Köbler, G. Rattan, and O. Verbitsky. On the power of color refinement. In *FCT*, pages 339–350, 2015.

W. Azizian and M. Lelarge. Characterizing the expressive power of invariant and equivariant graph neural networks. In *ICLR*, 2021.

L. Babai and L. Kucera. Canonical labelling of graphs in linear average time. In *FOCS*, 1979.

P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. In *COLT*, 2001.

P. L. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *NeurIPS*, 2017.

B. Bevilacqua, Y. Zhou, and B. Ribeiro. Size-invariant graph representations for graph classification extrapolations. In *ICML*, 2021.

M. Bianchini and F. Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8):1553–1565, 2014.

J. Böker. Graph similarity and homomorphism densities. In N. Bansal, E. Merelli, and J. Worrell, editors, *ICALP*, 2021.

A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NeurIPS*, 2013.

K. M. Borgwardt, M. E. Ghisu, F. Llinares-López, L. O'Bray, and B. Rieck. Graph kernels: State-of-the-art and future challenges. *Foundations and Trends in Machine Learning*, 13 (5–6), 2020.

S. Boucheron, G. Lugosi, and O. Bousquet. Concentration inequalities. In *Advanced Lectures on Machine Learning, ML Summer Schools 2003*, 2003.

G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint*, 2020.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 06 2002.

O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. In *Advanced Lectures on Machine Learning, ML Summer Schools 2003*, 2003.

J. Böker, R. Levie, N. Huang, S. Villar, and C. Morris. Fine-grained expressivity of graph neural networks. In *NeurIPS*, 2023.

J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identifications. *Combinatorica*, 12(4):389–410, 1992.

Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris, and P. Veličković. Combinatorial optimization and reasoning with graph neural networks. In *IJCAI*, 2021.

M. Chen, X. Li, and T. Zhao. On generalization bounds of a family of recurrent neural networks. In *AISTATS*, 2020.

Z. Chen, S. Villar, L. Chen, and J. Bruna. On the equivalence between graph isomorphism testing and function approximation with GNNs. In *NeurIPS*, pages 15868–15876, 2019.

Y. Chu and M. Raginsky. A unified framework for information-theoretic generalization bounds. In *NeurIPS*, 2023.

C.-Y. Chuang and S. Jegelka. Tree mover's distance: Bridging graph metrics and stability of graph neural networks. *NeurIPS*, 2022.

C. Y. Chuang, Y. Mroueh, K. Greenewald, A. Torralba, and S. Jegelka. Measuring generalization with optimal transport. In *NeurIPS*, volume 10, 2021.

C. Chung, J. Lee, and J. J. Whang. Representation learning on hyper-relational and numeric knowledge graphs with transformers. In *KDD*, 2023.

W. Cong, M. Ramezani, and M. Mahdavi. On provable benefits of depth in training graph convolutional networks. In *NeurIPS*, 2021.

G. Corso, H. Stark, S. Jegelka, T. Jaakkola, and R. Barzilay. Graph neural networks. *Nature Reviews Methods Primers*, 4(1):17, 2024.

H. Dai, B. Dai, and L. Song. Discriminative embeddings of latent variable models for structured data. In *ICLR*, 2016.

N. Daniëls and F. Geerts. A note on the VC dimension of 1-dimensional GNNs. *arXiv preprint*, 2024.

G. A. D'Inverno, M. Bianchini, and F. Scarselli. VC dimension of graph neural networks with Pfaffian activation functions. *arXiv preprint*, 2024.

O. Duranthon and L. Zdeborová. Asymptotic generalization error of a single-layer graph convolutional network. *arXiv preprint*, 2024.

A. Duval, S. V. Mathis, C. K. Joshi, V. Schmidt, S. Miret, F. D. Malliaros, T. Cohen, P. Lio, Y. Bengio, and M. M. Bronstein. A hitchhiker's guide to geometric GNNs for 3D atomic systems. *arXiv preprint*, 2023.

D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World.* Cambridge University Press, 2010.

R. El-Yaniv and D. Pechyony. Transductive rademacher complexity and its applications. In *COLT*, 2007.

P. M. Esser, L. C. Vankadara, and D. Ghoshdastidar. Learning theory can (sometimes) explain generalisation in graph neural networks. In *NeurIPS*, 2021.

J. Feng, Y. Chen, F. Li, A. Sarkar, and M. Zhang. How powerful are k-hop message passing graph neural networks. In *NeurIPS*, 2022.

B. J. Franks, C. Morris, A. Velingker, and F. Geerts. Weisfeiler–leman at the margin: When more expressivity matters. *arXiv preprint*, 2023.

A. M. Frieze and R. Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19:175–220, 1999.

V. K. Garg, S. Jegelka, and T. S. Jaakkola. Generalization and representational limits of graph neural networks. In *ICML*, 2020.

M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *NeurIPS*, 2019.

F. Geerts and J. L. Reutter. Expressiveness and approximation properties of graph neural networks. In *ICLR*, 2022.

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *ICLR*, 2017.

O. Goldreich. Introduction to testing graph properties. In *Property Testing.* Springer, 2010.

M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *IJCNN*, 2005.

M. Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory.* Cambridge University Press, 2017.

M. Grohe. The logic of graph neural networks. In *LICS*, 2021.

S. Gui, M. Liu, X. Li, Y. Luo, and S. Ji. Joint learning of label and environment causal independence for graph out-of-distribution generalization. In *NeurIPS*, 2023.

W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. *NeurIPS*, 2017.

D. Herbst and S. Jegelka. Higher-order graphon neural networks: Approximation and cut distance. In *Int. Conf. on Learning Representations (ICLR)*, 2025.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58, 1963.

D. N. Hoover. Relations on probability spaces and arrays of random variables. *Princeton Institute for Advanced Study (preprint)*, 1979.

D. N. Hoover. Row-column exchangeability and a generalized model for probability. *Exchangeability in Probability and Statistics*, pages 281–291, 1982.

S. Jegelka. Theory of graph neural networks: Representation and learning. *arXiv preprint*, 2022.

W. Jin, R. Barzilay, and T. Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *ICML*, 2018.

W. Jin, K. Yang, R. Barzilay, and T. Jaakkola. Learning multimodal graph-to-graph translation for molecular optimization. *ICLR*, 2019.

H. Ju, D. Li, A. Sharma, and H. R. Zhang. Generalization in graph neural networks: Improved PAC-Bayesian bounds on graph diffusion. *arXiv preprint*, 2023.

R. Karczewski, A. H. Souza, and V. Garg. On the generalization of equivariant graph neural networks. In *ICML*, 2024.

M. Karpinski and A. Macintyre. Polynomial bounds for VC dimension of sigmoidal and general Pfaffian neural networks. *Journal of Computer and System Sciences*, 54(1):169–176, 1997.

K. Kawaguchi, Z. Deng, K. Luh, and J. Huang. Robustness implies generalization via data-dependent generalization bounds. In *ICML*, 2022.

N. Keriven, A. Bietti, and S. Vaiter. Convergence and stability of graph convolutional networks on large random graphs. In *NeurIPS*, 2020.

A. Khovanskiĭ. *Fewnomials*, volume 88 of *Translations of Mathematical Monographs*. American Mathematical Society, 1991.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

V. Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transaction on Information Theory*, 47(5):1902–1914, 2001.

N. M. Kriege, C. Morris, A. Rey, and C. Sohler. A property testing framework for the theoretical expressivity of graph kernels. In *IJCAI*, 2018.

N. M. Kriege, F. D. Johansson, and C. Morris. A survey on graph kernels. *Applied Network Science*, 5(1):6, 2020.

R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, O. Vinyals, J. Stott, A. Pritzel, S. Mohamed, and P. Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

J. Langford and J. Shawe-Taylor. PAC-Bayes & margins. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS*, 2002.

T. Le and S. Jegelka. Limits, approximation and size transferability for gnns on sparse graphs via graphops. In *NeurIPS*, 2023.

J. Lee, M. Hwang, and J. J. Whang. Pac-bayesian generalization bounds for knowledge graph representation learning. In *ICML*, 2024.

R. Levie. A graphon-signal analysis of graph neural networks. In *NeurIPS*, 2023.

R. Levie, E. Isufi, and G. Kutyniok. On the transferability of spectral graph filters. *arXiv preprint*, 2019.

R. Levie, W. Huang, L. Bucci, M. Bronstein, and G. Kutyniok. Transferability of spectral graph convolutional neural networks. *Journal of Machine Learning Research*, 22(272):1–59, 2021.

H. Li, M. Wang, S. Liu, P. Chen, and J. Xiong. Generalization guarantee of training graph convolutional networks with graph topology sampling. In *ICML*, 2022a.

H. Li, X. Wang, Z. Zhang, and W. Zhu. Out-of-distribution generalization on graphs: A survey. *arXiv preprint*, 2022b.

S. Li, F. Geerts, D. Kim, and Q. Wang. Towards bridging generalization and expressivity of graph neural networks. *arXiv preprint*, 2024a.

X. Li, S. Gui, Y. Luo, and S. Ji. Graph structure extrapolation for out-of-distribution generalization. In *ICML*, 2024b.

R. Liao, Y. Li, Y. Song, S. Wang, W. L. Hamilton, D. Duvenaud, R. Urtasun, and R. S. Zemel. Efficient graph generation with graph recurrent attention networks. In *NeurIPS*, 2019.

R. Liao, R. Urtasun, and R. S. Zemel. A PAC-Bayesian approach to generalization bounds for graph neural networks. In *ICML*, 2021.

Y. Lin. A note on margin-based loss functions in classification. *Statistics and Probability Letters*, 68, 2004.

C. Louizos, U. Shalit, J. M. Mooij, D. A. Sontag, R. S. Zemel, and M. Welling. Causal effect inference with deep latent-variable models. In *NeurIPS*, 2017.

L. M. Lovász and B. Szegedy. Szemerédi's lemma for the analyst. *GAFA Geometric And Functional Analysis*, 2007.

L. Lovász. *Large Networks and Graph Limits*. American Mathematical Society, 2012.

S. Lv. Generalization bounds for graph convolutional neural networks via rademacher complexity. *arXiv preprint*, 2021.

J. Ma, J. Deng, and Q. Mei. Subgroup generalization and fairness of graph neural networks. In *NeurIPS*, 2021.

T. Maehara and H. NT. A simple proof of the universality of invariant/equivariant graph neural networks. *arXiv preprint*, 2019.

H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman. Invariant and equivariant graph networks. In *ICLR*, 2019.

S. Maskey, Y. Lee, R. Levie, and G. Kutyniok. Generalization analysis of message passing neural networks on large random graphs. In *NeurIPS*, 2022.

S. Maskey, R. Levie, and G. Kutyniok. Transferability of graph neural networks: An extended graphon approach. *Applied and Computational Harmonic Analysis*, 63:48–83, 2023.

S. Maskey, G. Kutyniok, and R. Levie. Generalization bounds for message passing networks on mixture of graphons. *arXiv preprint*, 2024.

D. A. McAllester. PAC-Bayesian model averaging. In *COLT*, 1999.

D. A. McAllester. Simplified pac-bayesian margin bounds. In *COLT*, 2003.

C. McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics 1989*, pages 148–188. Cambridge University Press, Cambridge, 1989.

C. Merkwirth and T. Lengauer. Automatic generation of complementary descriptors with molecular graph networks. *Journal of Chemical Information and Modeling*, 45(5):1159–1168, 2005.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. The MIT Press, Cambridge, MA, 2018.

C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *AAAI*, 2019a.

C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. *AAAI*, 2019b.

C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint*, 2020a.

C. Morris, G. Rattan, and P. Mutzel. Weisfeiler and Leman go sparse: Towards higher-order graph embeddings. In *NeurIPS*, 2020b.

C. Morris, Y. L., H. Maron, B. Rieck, N. M. Kriege, M. Grohe, M. Fey, and K. Borgwardt. Weisfeiler and Leman go machine learning: The story so far. *arXiv preprint*, 2021.

C. Morris, F. Geerts, J. Tönshoff, and M. Grohe. WL meet VC. In *ICML*, 2023.

C. Morris, F. Frasca, N. Dym, H. Maron, İ. İ. Ceylan, R. Levie, D. Lim, M. M. Bronstein, M. Grohe, and S. Jegelka. Future directions in foundations of graph machine learning. *arXiv preprint*, 2024.

S. Mukherjee, P. Niyogi, T. Poggio, and R. Rifkin. Learning theory: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*, 25, 2006.

B. Neyshabur, S. Bhojanapalli, and N. Srebro. A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. *ICLR*, 2018.

K. Oono and T. Suzuki. Optimization and generalization analysis of transduction through gradient boosting and application to multi-scale graph neural networks. In *NeurIPS*, 2020.

D. Pechyony. *Theory and practice of transductive learning.* PhD thesis, Technion - Israel Institute of Technology, Israel, 2008.

P. Pellizzoni, T. Schulz, D. Chen, and K. M. Borgwardt. On the expressivity and sample complexity of node-individualized graph neural networks. In *NeurIPS*, 2024.

C. Qian, D. Chételat, and C. Morris. Exploring the power of graph neural networks in solving linear optimization problems. *arXiv preprint*, 2023.

L. Rauchwerger, S. Jegelka, and R. Levie. Generalization, expressivity, and universality of graph neural networks on attributed graphs. *arXiv preprint*, 2024.

L. Ruiz, L. F. O. Chamon, and A. Ribeiro. Graphon neural networks and the transferability of graph neural networks. In *NeurIPS*, 2020.

L. Ruiz, L. F. O. Chamon, and A. Ribeiro. Graphon signal processing. *IEEE Transactions on Signal Processing*, pages 4961–4976, 2021a.

L. Ruiz, Z. Wang, and A. Ribeiro. Graphon and graph neural network stability. In *ICASSP*, 2021b.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

F. Scarselli, A. C. Tsoi, and M. Hagenbuchner. The Vapnik–Chervonenkis dimension of graph and recursive neural networks. *Neural Networks*, 108:248–259, 2018.

C. Shi, L. Pan, H. Hu, and I. Dokmanić. Homophily modulates double descent generalization in graph convolution networks. *Proceedings of the National Academy of Sciences of the United States of America*, 121, 2024.

T. Sun and J. Lin. Pac-bayesian adversarially robust generalization bounds for graph neural network. *arXiv*, 2024.

A. Suzuki, A. Nitanda, T. Suzuki, J. Wang, F. Tian, and K. Yamanishi. Tight and fast generalization error bound of graph embedding in metric space. In *ICML*, 2023.

H. Tang and Y. Liu. Towards understanding generalization of graph neural networks. In *ICML*, 2023.

V. Vapnik. *Statistical learning theory.* Wiley, 1998.

V. Vapnik. *Estimation of Dependences Based on Empirical Data, Second Editiontion.* Springer, 2006.

V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.

V. N. Vapnik and A. Chervonenkis. A note on one class of perceptrons. *Avtomatika i Telemekhanika*, 24(6):937–945, 1964.

A. Vasileiou, B. Finkelshtein, F. Geerts, R. Levie, and C. Morris. Covered forest: Fine-grained generalization analysis of graph neural networks. *arXiv preprint*, 2024.

S. Verma and Z. Zhang. Stability and generalization of graph convolutional neural networks. In *KDD*, 2019.

U. von Luxburg and B. Schölkopf. *Statistical Learning Theory: Models, Concepts, and Results*, volume 10. Handbook of the History of Logic, 2011.

B. Weisfeiler. *On Construction and Identification of Graphs*. Springer, 1976.

B. Weisfeiler and A. Leman. The reduction of a graph to canonical form and the algebra which appears therein. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

F. Wong, E. J. Zheng, J. A. Valeri, N. M. Donghia, M. N. Anahtar, S. Omori, A. Li, A. Cubillos-Ruiz, A. Krishnan, W. Jin, A. L. Manson, J. Friedrichs, R. Helbig, B. Hajian, D. K. Fiejtek, F. F. Wagner, H. H. Soutter, A. M. Earl, J. M. Stokes, L. D. Renner, and J. J. Collins. Discovery of a structural class of antibiotics with explainable deep learning. *Nature*, 2023.

S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui. Graph neural networks in recommender systems: A survey. *ACM Computing Surveys*, 55(5):97:1–97:37, 2023a.

Y. Wu, A. Bojchevski, and H. Huang. Adversarial weight perturbation improves generalization in graph neural networks. *AAAI*, 2023b.

H. Xu and S. Mannor. Robustness and generalization. *Machine Learning*, 86(3):391–423, 2012.

K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICLR*, 2018.

K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *ICLR*, 2019a.

K. Xu, S. Jegelka, W. Hu, and J. Leskovec. How powerful are graph neural networks? *ICLR*, 2019b.

K. Xu, J. Li, M. Zhang, S. Du, K. Kawarabayashi, and S. Jegelka. What can neural networks reason about? In *ICLR*, 2020.

K. Xu, M. Zhang, J. Li, S. S. Du, K.-I. Kawarabayashi, and S. Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. In *ICLR*, 2021.

H. Ye, C. Xie, T. Cai, R. Li, Z. Li, and L. Wang. Towards a theoretical framework of out-of-distribution generalization. In *NeurIPS*, 2021.

G. Yehudai, E. Fetaya, E. A. Meirom, G. Chechik, and H. Maron. From local structures to size generalization in graph neural networks. In *ICML*, 2021.

B. Zhang, G. Feng, Y. Du, D. He, and L. Wang. A complete expressiveness hierarchy for subgraph GNNs via subgraph Weisfeiler-Lehman tests. *arXiv preprint*, 2023.

S. Zhang, M. Wang, S. Liu, P. Y. Chen, and J. Xiong. Fast learning of graph neural networks with guaranteed generalizability: One-hidden-layer case. In *ICML*, 2020.

X. Zhou and H. Wang. The generalization error of graph convolutional networks may enlarge with more layers. *Neurocomputing*, 424, 2021.

A. The 1-dimensional Weisfeiler–Leman algorithm

The 1-WL or *color refinement* is a well-studied heuristic for the graph isomorphism problem, originally proposed by Weisfeiler and Leman [1968].[7] Intuitively, the algorithm determines if two graphs are non-isomorphic by iteratively coloring or labeling vertices. Given an initial coloring or labeling of the vertices of both graphs, e.g., their degree or application-specific information, in each iteration, two vertices with the same label get different labels if the number of identically labeled neighbors is unequal. These labels induce a vertex partition, and the algorithm terminates when, after some iteration, the algorithm does not refine the current partition, i.e., when a *stable coloring* or *stable partition* is obtained. Then, if the number of vertices annotated with a specific label is different in both graphs, we can conclude that the two graphs are not isomorphic. It is easy to see that the algorithm cannot distinguish all non-isomorphic graphs [Cai et al., 1992]. However, it is a powerful heuristic that can successfully decide isomorphism for a broad class of graphs [Arvind et al., 2015, Babai and Kucera, 1979].

Formally, let $G = (V(G), E(G), \ell)$ be a labeled graph. In each iteration, $t > 0$, the 1-WL computes a vertex coloring $C_t^1 \colon V(G) \to \mathbb{N}$, depending on the coloring of the neighbors. That is, in iteration $t > 0$, we set

$$C_t^1(v) \coloneqq \mathsf{RELABEL}\Big(\big(C_{t-1}^1(v), \{\!\{C_{t-1}^1(u) \mid u \in N(v)\}\!\}\big)\Big),$$

for all vertices $v \in V(G)$, where $\mathsf{RELABEL}$ injectively maps the above pair to a unique natural number, which has not been used in previous iterations. In iteration 0, the coloring $C_0^1 \coloneqq \ell$ is used. To test whether two graphs G and H are non-isomorphic, we run the above algorithm in "parallel" on both graphs. If the two graphs have a different number of vertices colored $c \in \mathbb{N}$ at some iteration, the 1-WL *distinguishes* the graphs as non-isomorphic. Moreover, if the number of colors between two iterations, t and $(t+1)$, does not change, i.e., the cardinalities of the images of C_t^1 and C_{i+t}^1 are equal, or, equivalently,

$$C_t^1(v) = C_t^1(w) \iff C_{t+1}^1(v) = C_{t+1}^1(w),$$

for all vertices v and w in $V(G \,\dot\cup\, H)$, then the algorithm terminates. For such t, we define the *stable coloring* $C_\infty^1(v) = C_t^1(v)$, for $v \in V(G \,\dot\cup\, H)$. The stable coloring is reached after at most $\max\{|V(G)|, |V(H)|\}$ iterations [Grohe, 2017]. We define the *color complexity* of a graph G as the number of colors computed by the 1-WL after $|V(G)|$ iterations on G.

B. Pfaffian functions

Pfaffian functions have been first introduce in Khovanskiĭ [1991] to extend Bezout's theorem, stating that the number of complex solutions of a set of polynomial equations can be estimated

---

[7]Strictly speaking, the 1-WL and color refinement are two different algorithms. That is, the 1-WL considers neighbors and non-neighbors to update the coloring, resulting in a slightly higher expressive power when distinguishing vertices in a given graph; see Grohe [2021] for details. Following customs in the machine learning literature, we consider both algorithms to be equivalent.

based on their degree. For different works exploiting properties of Pfaffian functions, see [Karpinski and Macintyre, 1997, Bianchini and Scarselli, 2014].

A Pfaffian chain of order $\ell \geq 0$ and degree $\alpha \geq 1$, in an open domain $U \subset \mathbb{R}^n$, is a sequence of analytic functions $f_1, f_2, , \ldots, f_\ell$ over U, satisfying the differential equations

$$\frac{df_j(\boldsymbol{x})}{dx_i} = \sum_{1 \leq i \leq n} g_{ij}(\boldsymbol{x}, f_1(\boldsymbol{x}), \ldots, f_j(\boldsymbol{x})), \quad 1 \leq j \leq \ell.$$

Here, $g_{ij}(\boldsymbol{x}, y_1, \ldots, y_j)$ are polynomials in $\boldsymbol{x} \in U$ and $y_1, \ldots y_j \in \mathbb{R}$ of degree not exceeding α. A function $f(\boldsymbol{x}) = P(\boldsymbol{x}, f_1(\boldsymbol{x}), \ldots, f_\ell(\boldsymbol{x}))$ where $P(\boldsymbol{x}, y_1, \ldots, y_j)$ is a polynomial of degree not exceeding β, is called a *Pfaffian function of format* (α, β, ℓ). Most of the functions with continuous derivatives are *Pfaffian functions*. In particular, the *arcatangent*, *logistic sigmoid*, and the *hyperbolic tangent* are *Pfaffian functions*.

C. Graph convolutional neural networks

Let $G \in \mathcal{G}_{n,d}$ be an unlabeled graph with adjacency matrix $\boldsymbol{A}$, Laplacian $\boldsymbol{L}$, and node features $\boldsymbol{x}_v \in \mathbb{R}^d$, for $v \in V(G)$. For the K-class graph classification problem, we use the following matrix notation of an L-layer GCN,

$$H_k = \sigma(\boldsymbol{L}\boldsymbol{H}_{k-1}\boldsymbol{W}_k) \quad (k\text{-th Graph Convolution Layer})$$

$$H_L = \frac{1}{n}\mathbf{1}_n\boldsymbol{H}_{L-1}\boldsymbol{W}_L \quad (\text{Readout Layer}),$$

where, $k \in [L-1]$, $\boldsymbol{H}_k \in \mathbb{R}^{n \times h_k}$, $\mathbf{1}_n \in \mathbb{R}^{1 \times n}$ is the all-one vector, $\boldsymbol{W}_j$ is the weight matrix of the j-th layer, and $\boldsymbol{H}_0 \in \mathbb{R}^{n \times d}$ is the initial node representation matrix, having rows equal to the initial node features $\boldsymbol{x}_u$, for $u \in V(G)$.

D. Extended graphon-theory

Graphons are utilized to analyze the generalization capabilities of MPNNs, either as models of the data distribution or as analytic tools for generic data distributions. In the former approach, graphs are assumed to be randomly sampled from graphons. In the latter, the non-complete space of graphs is embedded into the compact metric space of graphons, leveraging compactness to derive generalization bounds or universal approximation results for arbitrary data distributions. In what follows, we present the most fundamental properties of graphons and their relevance to graph theory. For simplicity, we focus on the main results concerning the space of graphons rather than graphon signals. However, when necessary, we reference works that extend these results to graphon signals.

D.1. Basics on graphon theory

Here, we review the basics on graphon theory.

Definitions We begin by recalling the definition of a graphon, which is a symmetric, measurable function $W \colon [0,1]^2 \to [0,1]$. The set of all graphons is denoted by $\mathcal{W}$. Graphons generalize graphs in the following way. Every graph G of order n can be represented as a graphon W_G.

This is achieved by partitioning $[0,1]$ into n intervals $(I_v)_{v \in V(G)}$, each of measure $1/n$, and defining $W_G(x,y)$ for $x \in I_u$ and $y \in I_v$ to be one if $\{u,v\} \in E(G)$ and zero otherwise. We refer to W_G as the graphon induced by the graph G.

Message-passing graph neural networks on graphons For a graphon $W \in \mathcal{W}$, an L-layer MPNN initializes a feature $\boldsymbol{h}_x^{(0)} \coloneqq \varphi_0 \in \mathbb{R}^{d_0}$, for $x \in [0,1]$. Then, for each layer $t \in [L]$, the features $\boldsymbol{h}_x^{(t)} \colon [0,1] \to \mathbb{R}^{d_t}$ are computed iteratively. After L layers, a single graphon-level feature $\boldsymbol{h}_W \in \mathbb{R}^{1 \times d}$ is computed as follows,

$$\boldsymbol{h}_x^{(t)} \coloneqq \varphi_t\left(\int_{[0,1]} W(x,y)\boldsymbol{h}_y^{(t-1)}\, d\lambda(y)\right), \quad \text{and} \quad \boldsymbol{h}_W \coloneqq \psi\left(\int_{[0,1]} \boldsymbol{h}_x^{(L)}\, d\lambda(x)\right).$$

Here, $(\varphi_t)_{t=1}^L$ and ψ are the functions defined in Equation (4). This definition generalizes architectures following Equation (4) to graphons. Specifically, we have the following result:

Lemma 29 ([Böker et al., 2023, Theorem 9]). Let $G \in \mathcal{G}_n$ for some $n \in \mathbb{N}$, let $(\varphi_t)_{t=1}^L$ define an L-layer MPNN model, and let ψ be Lipschitz. Then,

$$\boldsymbol{h}_G = \boldsymbol{h}_{W_G}. \qquad \qquad \square$$

The above definition of MPNNs (and Lemma 29) can be extended to graphon-signals $(W, \boldsymbol{f})$ by initializing $\boldsymbol{h}_x^{(0)} = \boldsymbol{f}(x)$ for $x \in [0,1]$, as described in Rauchwerger et al. [2024].

D.2. Graphons as generative graph models

A graphon can serve as a generative model of graphs. Given a graphon W, a random graph of n nodes is generated by sampling uniformly and independently points $\{X_i\}_{i=1}^n$ from $[0,1]$, and connecting each pair X_i, X_j in probability $W(X_i, X_j)$ to obtain the edges of the graph. Most works consider graphs as generative models of graphs to model the data distributions. These include papers that study *transferability*. Here, one studies a setting where an MPNN is trained on a given graph, assumed to be sampled from a graphon, and the goal is to show that the repercussion of the MPNN on another graph, sampled from the same graphon, is close to the repercussion on the original graph. This aims at modeling the ability to transfer MPNNs between different graphs of the same "type" [Levie et al., 2021, Ruiz et al., 2021a, Keriven et al., 2020, Maskey et al., 2023, Ruiz et al., 2021b]. Other works [Maskey et al., 2022, 2024] use graphons to model the data distribution in graph classification or regression settings. Here, one assumes that there is a finite set of template graphons, such that each graph from the dataset is sampled independently from one of these graphons; see [Maskey et al., 2024] for more details.

D.3. Graphons as the completion of the space of graphs

When analyzing properties like generalization or universal approximation of a model, one usually has to choose a metric or a topology for the input space. The compactness of the input space is a requirement both for using the Stone–Weierstrass theorem for universality analysis and for proving generalization bounds via a covering number analysis. In the context of graphs, when the size or degree n is not assumed to be bounded from above, the space of graphs is typically non-complete (and hence non-compact) concerning standard graph metrics. A standard tool for obtaining a complete space from a separable space is *completion*. For example, the real

numbers—a complete space—is the completion of the rationals—a non-complete space. In the context of graphs, as explained above, the space of graphons is the completion of the space of graphs concerning the cut distance. Hence, the universality and generalization analysis are developed for the "nice" compact space of graphons, and the results are then also valid for graphs as a special case. Such an analysis is presented in Levie [2023] for generalization and in Böker et al. [2023] for expressivity and universal approximation analysis.

Cut distance   For any $W \in L^2[0,1]^2$, we define the cut norm of W as

$$\|W\|_\square = \sup_{U,V \subset [0,1]} \Big| \int_{U \times V} W(x,y) dx dy \Big|$$

where $U, V \subset [0,1]$ are measurable. The cut metric between two graphons $W, W' \in \mathcal{W}$ is defined to be

$$d_\square(W, W') = \|W - W'\|_\square.$$

This is interpreted as the difference between the average edge weights of W and W' on the cut $U \times V$ that maximizes this difference.

Let $W' = W_{G'}$ be an SBM induced from a graph of m nodes, and W_G induced from a graph of $n = jm$ nodes, where $m, j \in \mathbb{N}$. In this case, $d_\square(W_G, W_{G'})$ measures how much the statistics/densities of the edge weights of G behave like the expected number of edges of graphs randomly sampled from the SBM $W_{G'}$. Hence, the cut metric has a pseudo-probabilistic interpretation in this case. How much the deterministic graph G looks like it was randomly sampled from the SBM G'. This interpretation can be extended to any pair of graphs G, G', where now $d_\square(W_G, W_{G'})$ is interpreted as how much G and G' look like they were sampled from the same SBM.

The cut metric is sensitive to the nodes' specific "ordering" in $[0,1]$. To obtain a "permutation invariant" metric, one defines the *cut distance*

$$\delta_\square(W', W) = \inf_{\psi \in S_{[0,1]}} d_\square(W', W^\psi),$$

where $S_{[0,1]}$ is the space of all measure preserving bijections – the analogue to permutations in general measure spaces.

The cut distance is a pseudo-metric. By introducing an equivalence relation $W \sim W'$ iff $\delta_\square(W, W') = 0$, and the quotient space $\widetilde{\mathcal{W}} := \mathcal{W}/\sim$, the space $\widetilde{\mathcal{W}}$ is a metric space with a metric $\delta_\square$ defined by $\delta_\square([W], [W']) = \delta_\square(W, W')$ where $W \in [W]$ and $W' \in [W']$ are two arbitrary representatives. By abuse of terminology, we call elements of $\widetilde{\mathcal{W}}$ also graphons.

The weak regularity lemma   One can extend the above interpretation of cut metric to distance between general graphons and SBMs. Namely, $d_\square(W, W_{G_n})$ quantifies how much the graphon W looks like it was randomly sampled from the SBM G_n of n nodes. The *weak regularity lemma* [Frieze and Kannan, 1999, Lovász and Szegedy, 2007] states that for every graphon W there is some SBM W_{G_n} such that $d_\square(W, W_{G_n}) \leq \frac{1}{\sqrt{c \log(n)}}$, where $c > 0$ is a universal constant. This means that the number of required blocks n^2 for approximating graphons by SBMs depends only on the desired error tolerance and not on any property of the graphon W. Hence, any graph looks like it was sampled from a SBM, where the number of blocks of the SBM does not depend on the graphon.

As a corollary of the weak regularity lemma, one can show that $\widetilde{\mathcal{W}}$ is a compact space undercut distance [Lovász and Szegedy, 2007]. Moreover, it follows that the space of graphs, embedded in $\widetilde{\mathcal{W}}$ via induced graphons, is dense in $\widetilde{\mathcal{W}}$. Hence, completing the space of graphs with cut distance is the compact space $\widetilde{\mathcal{W}}$.

E. Tree Mover's distance

The following introduces the *Tree Mover's Distance* (TMD), defined through the optimal transport problem. Following the notation from the original paper by Chuang and Jegelka [2022], we introduce the optimal transport problem and the Wasserstein distance. We then define an optimal transport problem between the sets of unrolled computation trees corresponding to the vertices of two attributed graphs. Next, we define a distance between labeled rooted trees and extend this definition to the space of attributed graphs, using the optimal transport problem and the unrolling trees of graphs' vertices.

The optimal transport problem   We begin with a brief introduction to *optimal transport* (OT) and the *Wasserstein distance*. Let $X = \{x_i\}_{i=1}^m$ and $Y = \{y_j\}_{j=1}^m$ be two multisets of m elements each. Let $\boldsymbol{C} \in \mathbb{R}^{m \times m}$ be the transportation cost for each pair, i.e., $C_{ij} = d(x_i, y_j)$, where d is a distance function between X and Y. The Wasserstein distance is defined through the following minimization problem,

$$\mathrm{OT}_d^*(X,Y) \coloneqq \min_{\boldsymbol{\gamma} \in \varGamma(X,Y)} \frac{\langle \boldsymbol{C}, \boldsymbol{\gamma} \rangle}{m}, \quad \varGamma(X,Y) = \{\gamma \in \mathbb{R}_+^{m \times m} \mid \boldsymbol{\gamma}\mathbf{1}_m = \boldsymbol{\gamma}^\top \mathbf{1}_m = \mathbf{1}_m\},$$

where $\langle \cdot, \cdot \rangle$ is defined as the sum of the elements resulting from the entry-wise multiplication of two matrices with equal dimensions, and $\mathbf{1}_m$ denotes the all ones m-dimensional column vector. In the following, we use the unnormalized version of the Wasserstein distance,

$$\mathrm{OT}_d(X,Y) \coloneqq \min_{\boldsymbol{\gamma} \in \varGamma(X,Y)} \langle \boldsymbol{C}, \boldsymbol{\gamma} \rangle = m \cdot \mathrm{OT}_d^*(X,Y).$$

Distance between rooted trees via hierarchical OT   Let (T, r) denote a rooted tree. We further let $\mathcal{T}_v$ be the multiset of subtrees of T consisting of subtrees rooted at the children of v. Determining whether two trees are similar requires iteratively examining whether the subtrees in each level are similar. By recursively computing the optimal transportation cost between their subtrees, we define the distance between two rooted trees $(T_a, r_a), (T_b, r_b)$. However, the number of subtrees could be different for r_a and r_b, i.e., $|\mathcal{T}_{r_a}| \neq |\mathcal{T}_{r_b}|$. To compute the OT between sets with different sizes, we augment the smaller set with *blank trees*.

Definition 30. A blank tree T_0 is a tree (graph) that contains a single vertex and no edge, where the vertex feature is the zero vector $\mathbf{0}_p \in \mathbb{R}^p$, and T_0^n denotes a multiset of n blank trees.

Definition 31. Given two multisets of trees $\mathcal{T}_u, \mathcal{T}_v$, define ρ to be a function that augments a pair of trees with blank trees as follows,

$$\rho \colon (\mathcal{T}_v, \mathcal{T}_u) \to \left(\mathcal{T}_v \cup T_0^{\max(|\mathcal{T}_u| - |\mathcal{T}_v|, 0)}, \; \mathcal{T}_u \cup T_0^{\max(|\mathcal{T}_v| - |\mathcal{T}_u|, 0)} \right).$$

We now recursively define a distance between rooted trees using the optimal transport problem.

Definition 32. The distance between two trees rooted $(T_a, r_a), (T_b, r_b)$ is defined recursively as:
$$\mathrm{TD}_\omega(T_a, T_b) := \begin{cases} \|\boldsymbol{x}_{r_a} - \boldsymbol{x}_{r_b}\| + \omega(L) \cdot \mathrm{OT}_{\mathrm{TD}_w}(\rho(T_{r_a}, T_{r_b})) & \text{if } L > 1, \\ \|\boldsymbol{x}_{r_a} - \boldsymbol{x}_{r_b}\| & \text{otherwise,} \end{cases}$$

where $\|\cdot\|$ is a norm on the feature space, $L = \max(\mathrm{Depth}(T_a), \mathrm{Depth}(T_b))$ and $\omega \colon \mathbb{N} \to \mathbb{R}^+$ is a depth-dependent weighting function.

Extension to attributed graphs   Below, we extend the above distance between rooted trees to the TMD on attributed graphs by calculating the optimal transportation cost between the graphs' unrolling trees.

Definition 33. Given two graphs G, H and $L > 0$, the tree mover's distance between G and H is defined as
$$\mathrm{TMD}_\omega^L(G, H) = \mathrm{OT}_{\mathrm{TD}_\omega}(\rho(\mathcal{T}_G^L, \mathcal{T}_H^L)),$$

where $\mathcal{T}_G^L$ and $\mathcal{T}_H^L$ are multisets of the depth-L unrolling trees of graphs G and H, respectively.

For completeness below we recall the defintion of the depth L unrolling of a labeled graph (G, ℓ_G) following Morris et al. [2020b]

Graph unrollings   Given an n-order labeled graph (G, ℓ_G), we define the *unrolling tree* of depth $L \in \mathbb{N}_0$ for a vertex $u \in V(G)$, denoted as $\mathrm{UNR}(G, u, L)$, inductively as follows.

1. For $L = 0$, we consider the trivial tree as an isolated vertex labeled $\ell_G(u)$.

2. For $L > 0$, we consider the root vertex with label $\ell_G(u)$ and, for $v \in N(u)$, we attach the subtree $\mathrm{UNR}(G, v, L - 1)$ under the root.

Note that above unrolling tree construction characterizes the 1-WL algorithm through the following lemma implying that the TMD distance is equivalent to 1-WL algorithm in terms of distinguishing non-isomorphic graphs.

Lemma 34 (Folklore, see, e.g., Morris et al. [2020a])**.** The following are equivalent for $L \in \mathbb{N}_0$, given a labeled graph (G, ℓ_G) and vertices $u, v \in V(G)$.

1. The vertices u and v have the same color after L iterations of the 1-WL.

2. The unrolling trees $\mathrm{UNR}(G, u, L)$ and $\mathrm{UNR}(G, v, L)$ are isomorphic.