Understanding Generalization in Node and Link Prediction

Antonis Vasileiou[1] Timo Stoll[1] Christopher Morris[1]

[1]Computer Science Department, RWTH Aachen University, Germany
{antonis.vasileiou, timo.stoll}@log.rwth-aachen.de

Abstract

Using message-passing graph neural networks (MPNNs) for node and link prediction is crucial in various scientific and industrial domains, which has led to the development of diverse MPNN architectures. Besides working well in practical settings, their ability to generalize beyond the training set remains poorly understood. While some studies have explored MPNNs' generalization in graph-level prediction tasks, much less attention has been given to node- and link-level predictions. Existing works often rely on unrealistic i.i.d. assumptions, overlooking possible correlations between nodes or links, and assuming fixed aggregation and impractical loss functions while neglecting the influence of graph structure. In this work, we introduce a unified framework to analyze the generalization properties of MPNNs in inductive and transductive node and link prediction settings, incorporating diverse architectural parameters and loss functions and quantifying the influence of graph structure. Additionally, our proposed generalization framework can be applied beyond graphs to any classification task under the inductive or transductive setting. Our empirical study supports our theoretical insights, deepening our understanding of MPNNs' generalization capabilities in these tasks.

1 Introduction

Graphs model interactions in 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], motivating machine learning methods for graph-structured data. Neural networks tailored to such data, mainly *message-passing graph neural networks* (MPNNs)[Gilmer et al., 2017, Scarselli et al., 2009], have gained wide attention, showing strong results in drug design[Wong et al., 2023], social network analysis [Borisyuk et al., 2024], weather forecasting [Lam et al., 2023], and combinatorial optimization [Cappart et al., 2021, Gasse et al., 2019, Scavuzzo et al., 2024, Qian et al., 2023].

MPNNs support node-, link-, and graph-level prediction [Chami et al., 2020]. While their generalization in graph-level tasks is well studied [Franks et al., 2023, Morris et al., 2023, Scarselli et al., 2018, Levie, 2023, Vasileiou et al., 2024a], node- and link-level generalization remains underexplored [Morris et al., 2024]. Existing node-level studies [Garg et al., 2020, Esser et al., 2021, Liao et al., 2021, Scarselli et al., 2018, Tang and Liu, 2023, Verma and Zhang, 2019] often assume i.i.d. samples, ignoring correlations from graph structure, and rely on restrictive losses (e.g., margin or 0-1), impractical for training. Likewise, despite advances in link-prediction models [Ali et al., 2022, Ye et al., 2022, Zhang et al., 2021], their generalization is unstudied.

Most analyses target either the *inductive* or *transductive* regime (see Section 2.2). In inductive learning, models train on multiple labeled graphs and predict on unseen ones, while transductive learning assumes a single graph with partially observed labels. Yet a unified understanding of MPNN generalization across these regimes is still missing. See Appendix A for a detailed discussion of related work.

Present work We introduce a unified framework for analyzing MPNN generalization in node- and link-prediction, extending recent covering number bounds [Vasileiou et al., 2024a]. Unlike Vasileiou et al. [2024a], we handle non-i.i.d. samples in these regimes, requiring non-trivial extensions. Concretely,

1. we present a unified framework covering most architectures for node- and link-prediction via generalized MPNNs (Section 3), including modern models [Zhang et al., 2021, Zhu et al., 2021].
2. We define pseudometrics (Section 3) that capture MPNN computations, satisfy a Lipschitz property, and reflect graph structure.
3. Using these pseudometrics, we derive generalization bounds for node- and link-prediction (Theorem 3, Theorem 4), accounting for sample dependencies in both inductive and transductive settings. Our bounds apply to standard losses such as cross-entropy.
4. Empirically, we show our theory aligns with practice, yielding a sharper understanding of when MPNNs generalize in node- and link-level tasks.

2 Background

In the following, we present the MPNN architecture used throughout this work and the two statistical settings (inductive and transductive) on which our generalization results are based.

2.1 Message-passing neural networks

A well-known and widely used class of graph-based models is the family of message-passing neural networks (MPNNs) [Gilmer et al., 2017]. These architectures learn vector representations for each node by iteratively aggregating information from their neighbors.

Sum aggregation For our analysis, we focus on a simplified yet expressive MPNN architecture (matching 1-WL expressivity [Morris et al., 2019]) that uses sum aggregation. Given a node-featured graph (G, a_G), we initialize node features as $\boldsymbol{h}_G(v)^{(0)} = a_G(v)$ for all $v \in V(G)$ and update them layer-wise by

$$\boldsymbol{h}_G^{(t)}(v) \coloneqq \varphi_t\Big(\boldsymbol{W}_t^{(1)}\boldsymbol{h}_G^{(t-1)}(v) + \boldsymbol{W}_t^{(2)} \sum_{u \in N(v)} \boldsymbol{h}_G^{(t-1)}(u)\Big), \tag{1}$$

for $v \in V(G)$, where $\varphi_t \colon \mathbb{R}^{d_{t-1}} \to \mathbb{R}^{d_t}$ is an L_{φ_t}-Lipschitz continuous function with respect to the metric induced by the 2-norm, for $d_t \in \mathbb{N}$ and $t \in [L]$. The matrices $\boldsymbol{W}_t^{(1)}, \boldsymbol{W}_t^{(2)} \in \mathbb{R}^{d_{t-1} \times d_t}$ are assumed to have their 2-norm bounded by some constant $B > 0$.

MPNNs for link prediction Given a graph, MPNNs for link prediction estimate the probability of a link between two nodes. Early models such as RGCN [Schlichtkrull et al., 2018a], CompGCN [Vashishth et al., 2020a], and GEM-GCN [Yu et al., 2020] relied on node embeddings combined into pair representations, but Zhang et al. [2021] showed these are insufficient to capture node interactions. This motivated more expressive architectures, including SEAL, C-MPNNs, and NCNs (Appendix F).

2.2 Inductive and transductive learning

This section presents the two learning frameworks underlying our generalization analysis: *inductive* and *transductive learning*.

Inductive setting Let $\mathcal{X}$ denote the input space and $\mathcal{Y}$ the label space, and define $\mathcal{Z} \coloneqq \mathcal{X} \times \mathcal{Y}$. A learning algorithm $\mathcal{A}$ receives as input a training sample $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^N \subset \mathcal{Z}$ drawn i.i.d. from an unknown distribution μ on $\mathcal{Z}$, and outputs a hypothesis $h = \mathcal{A}_\mathcal{S} \in \mathcal{H}$, where $\mathcal{H}$ is a hypothesis class. Given a bounded loss function $\ell \colon \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$, the expected risk and the empirical risk are defined as

$$\ell_{\text{exp}}(\mathcal{A}_\mathcal{S}) \coloneqq \mathbb{E}_{(x,y)\sim\mu}[\ell(\mathcal{A}_\mathcal{S}, x, y)], \quad \text{and} \quad \ell_{\text{emp}}(\mathcal{A}_\mathcal{S}) \coloneqq \frac{1}{N} \sum_{(x,y) \in \mathcal{S}} \ell(\mathcal{A}_\mathcal{S}, x, y),$$

respectively. The generalization error is then defined as $|\ell_{\text{emp}}(\mathcal{A}_\mathcal{S}) - \ell_{\text{exp}}(\mathcal{A}_\mathcal{S})|$.

Transductive setting In contrast to the inductive case, the transductive setting does not assume a distribution over $\mathcal{Z}$. Instead, we are given a fixed dataset $\mathcal{Z} = \{(x_i, y_i)\}_{i=1}^{m+u}$ of m labeled and u unlabeled examples. A transductive learning algorithm receives the full input set $\{x_i\}_{i=1}^{m+u}$ and the labels $\{y_i\}_{i=1}^m$ of a randomly selected training subset. The goal is to minimize the average loss over the remaining u test points. Following Vapnik [2006, Setting 1] for the model sampling process we let π be a random permutation over $[m+u]$, and we define,

$$R_m \coloneqq \frac{1}{m}\sum_{i=1}^{m} \ell(\mathcal{A}_\pi, x_{\pi(i)}, y_{\pi(i)}), \quad \text{and } R_u \coloneqq \frac{1}{u}\sum_{i=m+1}^{m+u} \ell(\mathcal{A}_\pi, x_{\pi(i)}, y_{\pi(i)}).$$

The permutation π is treated as a random variable, uniformly distributed over all permutations of $[m+u]$, and models the process in which the m training samples are drawn uniformly at random *without* replacement from the $m+u$ elements of $\mathcal{Z}$. The generalization error is given by $|R_m - R_u|$. Further formal details, including the permutation model, symmetry assumptions, and loss function, are provided in Appendix H. In the remainder of the paper, we focus on generalization bounds for node and link prediction under binary classification.

3 Generalized MPNNs and unrolling distances

We present a unified framework for generalized MPNNs that captures both node- and edge-level tasks. At a high level, a *representation task* (Appendix E) specifies the entities (nodes, links, or graphs) for which vector representations are computed. Formally, it consists of pairs $(G, S) \in \{(G, S) \mid G \in \mathcal{G}', S \subseteq V(G)\}$, where $\mathcal{G}'$ is a family of graphs and S is the relevant subset of nodes (e.g., a single node, a pair for a link, or a larger set). We further introduce pseudometrics (unrolling distances) between nodes or edges that satisfy a Lipschitz continuity property for generalized MPNNs.

Generalized MPNNs In link prediction, MPNNs are often applied to transformed graphs to encode structural information (e.g., the labeling trick [Zhang et al., 2021]). To unify graph-, node-, and link-level prediction tasks, we introduce *generalized MPNNs*. They operate in three steps: (i) transform the input graph with $\mathcal{T}$ and apply an MPNN to obtain node embeddings, (ii) use a selection function $\mathcal{V}$ to identify nodes relevant to each target, and (iii) aggregate these embeddings with a pooling function Ψ to compute the final representation. The resulting model is denoted by $\boldsymbol{h}^{(L)}_{\mathcal{T},\mathcal{V},\Psi}(G, S) \in \mathbb{R}^d$ for (G, S) in some representation task $(\mathcal{G}', \mathfrak{R})$. Formal definitions of representation tasks and the full construction of $(\mathcal{T}, \mathcal{V}, \Psi)$-$\mathsf{MPNN}(L)$ are given in Appendix E. For example, standard node prediction MPNNs (Equation (1)) arise by setting $\mathcal{T}(G) = G$, $\mathcal{V}(G, u) = \{u\}$, and $\Psi(F(G, \{u\})) = \boldsymbol{h}^{(L)}_G(u)$. Likewise, link prediction models such as SEAL, C-MPNNs, and NCNs fit naturally into this framework (Appendix F.1). Since pooling is central, we highlight sum-pooling and introduce the more general *sub-sum* property (Definition 15), satisfied by all architectures considered and essential for our theoretical results (Proposition 16).

Unrolling-based distances Unrolling distances quantify the similarity between the computation trees (or unrollings) induced by generalized MPNNs. Given a graph-representation task $(\mathcal{G}', \mathfrak{R})$, a transformation $\mathcal{T}$, a selection $\mathcal{V}$, and a depth L, each input (G, S) is associated with a multiset of unrolling trees $\mathcal{F}^{(L)}(G, S)$, which describe how information propagates from the selected nodes up to L layers. These trees are closely related to the computation process of message passing and to the 1-WL algorithm. The unrolling distance $\mathrm{UD}^{(L)}_{\mathcal{T},\mathcal{V}}$ between two inputs (G_1, S_1) and (G_2, S_2) is defined by comparing their unrolling forests. Intuitively, the distance measures the minimal feature mismatch between nodes under edge-preserving bijections, while allowing for a padding procedure that handles different forest sizes. This construction yields a well-defined pseudo-metric tailored to $(\mathcal{T}, \mathcal{V})$-MPNNs. Crucially, this pseudo-metric aligns with the Lipschitz continuity of generalized MPNNs: if the pooling function satisfies the sub-sum property, then the output representations of the model change at most proportionally to the unrolling distance. Formal definitions, the padding procedure, and full proofs are presented in Appendix E.

4 Robustness generalization under dependency

Xu et al. [2021] derived generalization bounds for the inductive setting under the i.i.d. assumption, based on covering numbers with respect to a suitable pseudo-metric. In the following, we extend the robustness framework of Xu and Mannor [2012] to handle dependent data, providing generalization guarantees in both the inductive and transductive regimes. For a more detailed account of how we extend these theoretical results to data-dependent regimes and reformulate the robustness property as a Lipschitz continuity condition, we refer to Appendix J and Appendix K.

Inductive robustness generalization analysis Independence plays a central role in deriving generalization bounds, particularly in the proof of Theorem 3 ([Xu and Mannor, 2012]), which relies on applying the Bretagnolle–Huber–Carol inequality to a multinomial vector drawn from an i.i.d. sample [A. W. van der Vaart, 1996, Proposition A.6.6]. Since this inequality ultimately depends on Hoeffding's inequality, the i.i.d. assumption is essential.

To relax this assumption, we follow Janson [2004], who introduced *dependency graphs*, graph structures in which nodes represent random variables and edges indicate potential dependencies, to extend classical concentration bounds to the dependent setting. We formally define dependency graphs in Appendix K.1 and prove Lemma 28, which is used to derive the following generalization result.

Theorem 1. Let $(\mathcal{X}, d_{\mathcal{X}})$ and $(\mathcal{Y}, d_{\mathcal{Y}})$ be (pseudo)metric spaces. Let $\mathcal{A}$ be a learning algorithm with hypothesis class $\mathcal{H} \subset \mathcal{X}^{\mathcal{Y}}$ consisting of $C_{\mathcal{H}}$-Lipschitz functions, and let ℓ be a C_ℓ-Lipschitz loss function bounded by $M > 0$. Then, for any $\delta > 0$, with probability (over the sample) at least $1 - \delta$, the following holds for all samples $\mathcal{S}$ with dependency graph $G[\mathcal{S}]$:

$$\left|\ell_{\exp}(\mathcal{A}_{\mathcal{S}}) - \ell_{\mathrm{emp}}(\mathcal{A}_{\mathcal{S}})\right| \leq C\varepsilon + M\sqrt{\frac{\chi(G[\mathcal{S}])(2(K_\varepsilon + 1)\log 2 + 2\log(1/\delta))}{|\mathcal{S}|}}, \quad \forall \varepsilon > 0,$$

where $K_\varepsilon = \mathcal{N}(\mathcal{Z}, d_\infty, \varepsilon)$ is the covering number of the space $(\mathcal{Z}, d_\infty)$ with radius ε, $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, $d_\infty \coloneqq \max\{d_{\mathcal{X}}, d_{\mathcal{Y}}\}$, $\chi(G[\mathcal{S}])$ is the chromatic number of the dependency graph $G[\mathcal{S}]$, and C is a constant depending linearly on $C_{\mathcal{H}}$ and C_ℓ.

Transductive robustness generalization analysis Below, we derive generalization bounds in the transductive setting using robustness. Unlike the inductive case, where robustness alone suffices, we require that the learning algorithm be stable to small changes in the training set. Intuitively, a transductive algorithm is β-stable if swapping one training and one test input changes the model's predictions by at most β. The formal definition is provided in Appendix K.2.

We now derive generalization bounds for robust and stable transductive learning algorithms. In this setting, Azuma's inequality [Azuma, 1967] replaces Hoeffding's inequality (see Appendix D, Theorem 13), and we extend the Bretagnolle–Huber–Carol inequality to Lemma 32 leading to the following generalization result for the transductive setting.

Theorem 2. Let $(\mathcal{X}, d_{\mathcal{X}})$ and $(\mathcal{Y}, d_{\mathcal{Y}})$ be (pseudo)metric spaces, and let $\mathcal{A}$ be a transductive learning algorithm on $\mathcal{Z} = \{z_i\}_{i=1}^{m+u}$, where $z_i \in \mathcal{X} \times \mathcal{Y}$ for all $i \in [m+u]$. Suppose that the hypothesis class $\mathcal{H}$ of $\mathcal{A}$ consists of $C_{\mathcal{H}}$-Lipschitz functions and that $\mathcal{A}$ satisfies uniform transductive stability with parameter $\beta > 0$. Assume further that the loss function ℓ is bounded by $M > 0$ and is C_ℓ-Lipschitz. Then, for any $\delta > 0$, with probability (over the sample) at least $1 - \delta$,

$$\left|R_m - R_u\right| \leq C\varepsilon + \left(\tfrac{1}{\sqrt{m}} + \tfrac{1}{\sqrt{u}}\right) \cdot M \cdot K_\varepsilon \sqrt{2(K_\varepsilon + 1)\log 2 + 2\log(1/\delta)} + C_\ell \beta,$$

where $K_\varepsilon = \mathcal{N}(\mathcal{Z}, d_\infty, \varepsilon)$ is the covering number of the space $(\mathcal{Z}, d_\infty)$ with radius ε, $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, $d_\infty \coloneqq \max\{d_{\mathcal{X}}, d_{\mathcal{Y}}\}$, and C is a constant depending linearly on $C_{\mathcal{H}}$ and C_ℓ.

5 Graph learning and robustness generalization

This section provides generalization bounds for graph representation learning tasks, focusing on node and link prediction under both the inductive and transductive frameworks, combining our main results Theorem 1 and Theorem 2 with the pseudo-metrics defined in Section 3.

Inductive node and link classification We now present generalization results for node and link prediction in the inductive setting using Theorem 1. Specifically, we consider binary classification with input space $\mathcal{X} \coloneqq \mathcal{G}_d^{\mathbb{R}} \otimes V$ (node prediction) or $\mathcal{X} \coloneqq \mathcal{G}_d^{\mathbb{R}} \otimes E$ (link prediction), and label space $\mathcal{Y} \coloneqq \{0, 1\}$. Our goal is to apply Theorem 1 to inductive learning algorithms by verifying the Lipschitz condition via unrolling distances.

In Section 3 and Proposition 20, we introduced a suitable pseudo-metric on $\mathcal{X}$ and proved its Lipschitz property. To extend this to $\mathcal{X} \times \mathcal{Y}$, we define a pseudo-metric d_∞ using the discrete metric δ on $\{0, 1\}$:

$$\delta(y_1, y_2) = \begin{cases} 0, & \text{if } y_1 = y_2, \\ \infty, & \text{otherwise,} \end{cases} \quad d_\infty((x, y), (x', y')) \coloneqq \max\{\mathrm{UD}_{\mathcal{T},\mathcal{V}}^{(L)}(x, x'), \delta(y, y')\}.$$

Data distribution dependency assumptions In node classification with multiple labeled graphs, it is reasonable to assume that nodes from different graphs are independent, as they are generated from separate structures and do not share edges or features. In contrast, nodes within the same graph may exhibit dependencies due to the relational structure of the graph. Formally, we assume:

(A) If (G_1, u_1, y_1) and (G_2, u_2, y_2) are two samples with $G_1 \neq G_2$, then they are independent; otherwise, they may be dependent.

We now present the main generalization result for binary node classification. The same result extends to the multi-class case and to link prediction under analogous assumptions.

Theorem 3 (Binary classification generalization)**.** Let $\mathcal{A}$ be a learning algorithm on $\mathcal{Z} = \mathcal{G}_d^{\mathbb{R}} \otimes V \times \{0,1\}$. Consider the hypothesis class $\mathcal{H}$ consisting of $(\mathcal{T}, \mathcal{V}, \Psi)$-MPNNs with L-layers, where Ψ is sub-sum. For a sample $\mathcal{S}$ (under the assumption (A)), and a loss function $\ell \colon \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$ that is bounded by some $M \in \mathbb{R}$, and $\ell(h, \cdot)$ being C_ℓ-Lipschitz concerning d_∞, we have,

$$|\ell_{\text{exp}}(\mathcal{A}_\mathcal{S}) - \ell_{\text{emp}}(\mathcal{A}_\mathcal{S})| \leq 2C\varepsilon + M\sqrt{\frac{2D_\mathcal{S}\big((2K_\varepsilon + 1)\log 2 + \log(\frac{1}{\delta})\big)}{N}}, \quad \text{for all } \varepsilon > 0,$$

where $K_\varepsilon = \mathcal{N}\Big(\mathcal{G}_d^{\mathbb{R}} \otimes V, \mathrm{UD}_{\mathcal{T},\mathcal{V}}^{(L)}, \varepsilon\Big)$, C is a constant depending on L, C_ℓ, on the Lipschitz constants L_{φ_t} of the MPNN layers, and $D_\mathcal{S}$ is the maximum number of samples in $\mathcal{S}$ that have been derived from the same graph, i.e., $D_\mathcal{S} \coloneqq \max_{G \in \mathcal{G}_d^{\mathbb{R}}} |\{(G, v, y) \in \mathcal{S} \mid v \in V(G), y \in \{0,1\}\}|$.

Note that the previous bound can become arbitrarily large when unbounded node degrees are present. However, unlike in graph classification, it is independent of the graph size. By restricting to graphs with maximum degree $q \in \mathbb{N}$, we can bound the covering number; see Corollary 34 in Appendix L.

Transductive node and link classification Similarly to the previous section, we present the node/link classification generalization result under the transductive setting utilizing Theorem 2.

Theorem 4 (Binary classification generalization)**.** Let $\mathcal{A}$ be a transductive learning algorithm satisfying β-uniform stability on $\mathcal{Z} = \{(x_i, y_i)\}_{i=1}^{m+u}$ where $(x_i, y_i) \in \mathcal{X} \times \{0,1\}$, and $\mathcal{X} = \mathcal{G}_d^{\mathbb{R}} \otimes V$, or $\mathcal{X} = \mathcal{G}_d^{\mathbb{R}} \otimes E$. Consider the hypothesis class $\mathcal{H}$ consisting of $(\mathcal{T}, \mathcal{V}, \Psi)$-MPNNs with L-layers, where Ψ is sub-sum. For a loss function $\ell \colon \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$ that is bounded by some $M \in \mathbb{R}$, and $\ell(h, \cdot)$ being C_ℓ-Lipschitz with respect to d_∞, we have, for $\varepsilon > 0$,

$$|\ell_{\text{exp}}(\mathcal{A}_\mathcal{S}) - \ell_{\text{emp}}(\mathcal{A}_\mathcal{S})| \leq 2C\varepsilon + M \cdot K_\varepsilon \cdot \left(\frac{1}{\sqrt{m}} + \frac{1}{\sqrt{u}}\right)\sqrt{2(2K_\varepsilon + 1)\log 2 + 2\log\left(\frac{1}{\delta}\right)} + C_\ell \beta,$$

where $K_\varepsilon = \mathcal{N}\Big(\mathcal{X}, \mathrm{UD}_{\mathcal{T},\mathcal{V}}^{(L)}, \varepsilon\Big)$, C is a constant depending on L, C_ℓ, and on Lipschitz constants L_{φ_t} of the message passing layers.

Intuitively, our main theorems (Theorem 3 and Theorem 4) show that MPNNs generalize better when the graph space is less complex, as measured by covering numbers under the unrolling distance: the smaller the covering number, the better the generalization performance.

6 Limitations and looking ahead

While our framework offers a principled approach to studying MPNN generalization, it has several limitations. Computing the proposed bounds—especially for link prediction—can be costly due to unrolling-based pseudo-metrics. The Lipschitz constant—central to the bounds—may be large for certain architectures, yielding loose estimates. Our analysis also requires pooling functions with the sub-sum property—excluding common choices like mean pooling and attention. Moreover, although the bounds cover both inductive and transductive settings, evaluating them in practice demands estimating all parameters in Theorems 3 and 4. Finally, as in generalization theory more broadly, the bounds may be vacuous in some regimes, since they govern the whole hypothesis class rather than trained models, which typically achieve much smaller losses. Still, such uniform bounds are useful for characterizing graph spaces where MPNNs can generalize.

Looking ahead, we plan to extend the framework to broader architectural choices—including alternative pooling and normalization—making it applicable to classes like graph transformers [Müller et al., 2023]. We also aim to connect our covering-number framework with SGD learning dynamics.

Table 1: Generalization results for different sampling strategies related to **Q2** for inductive node classification on the PATTERN dataset. Strategy *random* refers to training nodes sampled from a few graphs (resulting in higher sample dependency), while strategy *uniform* uses nodes sampled uniformly across many distinct graphs. Strategy *mixed* uniformly samples from a predefined number of graphs. $D_{\mathcal{S}}$ denotes the maximum sampled nodes of a single graph in the training set. Further, n_{train} and n_{test} denote the number of train and test nodes.

Method	**Mixed-4k-r**	**Mixed-4k-u**	**Mixed-8k-r**	**Mixed-8k-u**	**Random**	**Uniform**
n_{train}	120 000	120 000	120 000	120 000	120 000	120 000
n_{test}	116 232	116 232	116 232	116 232	116 232	116 232
$D_{\mathcal{S}}$	183	15	170	15	186	15
Training loss	0.440 ± 0.010	1.5806 ± 0.003	0.430 ± 0.011	1.579 ± 0.010	0.2485 ± 0.0073	1.5739 ± 0.0123
Test loss	1.763 ± 0.002	1.516 ± 0.030	1.749 ± 0.013	1.515 ± 0.034	1.8007 ± 0.0426	1.5114 ± 0.0152
Gen. gap	1.323	0.0646	1.319	0.064	1.552	0.0625

Table 2: Evaluation of node prediction bounds with $p = 0.001$, $cc = 1$ for LSP-ER(n, p, cc) graphs related to **Q3**, where n denotes the number of nodes, p the probability of an edge between nodes, and cc the number of random links between disconnected components. Further experiments are evaluated in Appendix M.

Dataset	**(100, 0.001, 1)**	**(200, 0.001, 1)**	**(500, 0.001, 1)**	**(1000, 0.001, 1)**
Calculated bound	6.58	6.07	5.67	2.26
Generalization gap	0.58	0.57	0.46	0.44

7 Experimental study

In the following, we investigate to what extent our theoretical results translate into practice. Specifically, we answer the following questions.

Q1 To what extent are MPNN outputs correlated with unrolling distances, and does this support the Lipschitz property empirically? Can we reliably estimate the corresponding Lipschitz constants?
Q2 Is assumption (A), used in Theorem 3, reasonable in the context of node-level prediction tasks? Specifically, does training on nodes from more distinct graphs lead to better generalization performance than training on nodes primarily drawn from a single graph?
Q3 Is the behavior of our theoretical bounds consistent with the actual generalization gap, defined as the difference between training and test error across different datasets?

Results and discussion We use a simplified version of GIN [Xu et al., 2019] aligned with Equation (1) for our experiments on node prediction tasks, and a simplified version of SEAL [Zhang and Chen, 2018] for link prediction tasks. To address **Q3**, we use synthetic datasets based on Erdős–Rényi generated graphs (LSP-ER) so that we can control and efficiently compute the covering number of the graph. See Appendix M for details on our synthetic datasets, neural architectures, experimental protocol, and model configurations. In the following, we address questions **Q1** to **Q3**.

Q1 In Figure 3, Figure 2, Figure 5, and Figure 4, we observe a correlation between the Euclidean norm of the difference in MPNN outputs and the corresponding unrolling distances. While some models may not exhibit a linear relationship, this does not contradict the Lipschitz property. Importantly, the Lipschitz constant (from Proposition 20) can be upper-bounded by the slope of the line passing through $(0, 0)$ that lies above all observed data points.

Q2 To validate results and assumptions in the inductive setting, where training involves nodes sampled from multiple graphs, and to test the alignment between empirical performance and the bounds given in Theorem 3, we use different sampling processes (Uniform, Random, Mixed-(u,r)). Each sampling process contains the same number of training and nodes, and the exact same test set, but differs in the number of distinct graphs used for sampling the training nodes. In Table 1, we observe that MPNNs tend to generalize significantly better when the training set contains nodes sampled from many distinct graphs (and hence smaller value of $D_{\mathcal{S}}$ in Theorem 3), compared to when nodes are drawn from only a few graphs (and hence larger $D_{\mathcal{S}}$) verifying our dependence assumptions in the inductive setting.

Q3 According to Theorem 10, a simpler graph (characterized by a smaller covering number K_{ε}) should result in a tighter generalization bound. This is exactly what we observe in Table 2, Table 5 and Table 6. More specifically, we increase the number of nodes n while keeping the edge probability fixed. This makes the graph structurally simpler. As predicted, both the theoretical bound and the empirical generalization gap decrease. Similar experiments can be found in Table 7, where we increase the edge probability while keeping the number of nodes fixed. This leads to a more complex graph, resulting in both a higher theoretical bound and a larger generalization gap.

8 Conclusion

In this work, we investigated how graph structure influences the generalization performance of MPNNs in non-i.i.d. settings, considering both inductive and transductive learning frameworks. We introduced a unified theoretical framework encompassing most state-of-the-art MPNN-based architectures for node and link prediction tasks. Our empirical study supports the theoretical findings and validates the underlying assumptions. *Overall, our theoretical framework constitutes an essential initial step in unraveling how graph structure influences the generalization abilities of MPNNs in node- and link-level prediction tasks. Beyond this, it provides a flexible framework for analyzing generalization in data-dependent settings, guided by the structural properties of the input space through the use of appropriately defined pseudo-metrics.*

References

J. A. W. A. W. van der Vaart. *Weak Convergence and Empirical Processes: With Applications to Statistics*. Springer, 1996.

M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, M. Galkin, S. Sharifzadeh, A. Fischer, V. Tresp, and J. Lehmann. Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12): 8825–8845, 2022.

J. Altschuler, J. Weed, and P. Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. In *Advances in Neural Information Processing Systems*, 2017.

V. Arvind, J. Köbler, G. Rattan, and O. Verbitsky. On the power of color refinement. In *International Symposium on Fundamentals of Computation Theory*, 2015.

K. Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, 19, 1967.

L. Babai and L. Kucera. Canonical labelling of graphs in linear average time. In *Symposium on Foundations of Computer Science*, 1979.

A. Baranwal, K. Fountoulakis, and A. Jagannath. Graph convolution for semi-supervised classification: Improved linear separability and out-of-distribution generalization. In *International Conference on Machine Learning*, 2021.

P. Barceló, M. Galkin, C. Morris, and M. A. R. Orth. Weisfeiler and Leman go relational. *arXiv preprint*, 2022.

P. L. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, 2017.

I. I. Baskin, V. A. Palyulin, and N. S. Zefirov. A neural device for searching direct arxiv preprintelations between structures and properties of chemical compounds. *Journal of Chemical Information and Computer Sciences*, 37(4):715–721, 1997.

J. Bento and S. Ioannidis. A family of tractable graph metrics. *Applied Network Science*, 4(1):107, 2019.

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.

F. Borisyuk, S. He, Y. Ouyang, M. Ramezani, P. Du, X. Hou, C. Jiang, N. Pasumarthy, P. Bannur, B. Tiwana, P. Liu, S. Dangi, D. Sun, Z. Pei, X. Shi, S. Zhu, Q. Shen, K.-H. Lee, D. Stein, B. Li, H. Wei, A. Ghoting, and S. Ghosh. Lignn: Graph neural networks at linkedin. *arXiv preprint*, 2024.

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

J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and deep locally connected networks on graphs. In *International Conference on Learning Representation*, 2014.

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 *International Joint Conference on Artificial Intelligence*, 2021.

B. P. Chamberlain, S. Shirobokov, E. Rossi, F. Frasca, T. Markovich, N. Y. Hammerla, M. M. Bronstein, and M. Hansmire. Graph neural networks for link prediction with subgraph sketching. In *International Conference on Learning Representations*, 2023.

I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *arXiv preprint*, 2020.

C.-Y. Chuang and S. Jegelka. Tree mover's distance: Bridging graph metrics and stability of graph neural networks. *Advances in Neural Information Processing Systems*, 2022.

C. Y. Chuang, Y. Mroueh, K. Greenewald, A. Torralba, and S. Jegelka. Measuring generalization with optimal transport. In *Advances in Neural Information Processing Systems*, 2021.

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

M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3837–3845, 2016.

G. A. D'Inverno, M. Bianchini, and F. Scarselli. VC dimension of graph neural networks with Pfaffian activation functions. *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. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, 2015.

V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson. Benchmarking graph neural networks. *J. Mach. Learn. Res. (JMLR)*, 24:43:1–43:48, 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 *Annual Conference on Learning Theory*, 2007.

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

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

F. Gama, A. G. Marques, G. Leus, and A. Ribeiro. Convolutional neural network architectures for signals supported on graphs. *IEEE Transactions on Signal Processing*, 67(4):1034–1049, 2019.

V. K. Garg, S. Jegelka, and T. S. Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, 2020.

M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2019.

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

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

C. Goller and A. Küchler. Learning task-dependent distributed representations by backpropagation through structure. In *International Conference on Neural Networks*, 1996.

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

M. Grohe. The logic of graph neural networks. In *Symposium on Logic in Computer Science*, 2021.

W. L. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

B. Hammer. Generalization ability of folding networks. *IEEE Trans. Knowl. Data Eng.*, pages 196–206, 2001.

R. V. Handel. Probability in high dimension. *Princeton.Edu*, 2014.

X. Huang, M. R. Orth, İsmail İlkan Ceylan, and P. Barceló. A theory of link prediction via relational Weisfeiler-Leman on knowledge graphs. In *Advances in Neural Information Processing Systems*, 2023.

S. Janson. Large deviations for sums of partly dependent random variables. *Random Structures & Algorithms*, 24, 2004.

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 *International Conference on Machine Learning*, 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.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

T. N. Kipf and M. Welling. Variational graph auto-encoders. *arXiv preprint*, 2016.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

D. B. Kireev. Chemnet: A novel neural network based method for graph/property mapping. *Journal of Chemical Information and Computer Sciences*, 35(2):175–180, 1995.

L. Kong, Y. Chen, and M. Zhang. Geodesic graph neural network for efficient graph representation learning. In *Advances in Neural Information Processing Systems*, 2022.

N. M. Kriege, C. Morris, A. Rey, and C. Sohler. A property testing framework for the theoretical expressivity of graph kernels. In *International Joint Conference on Artificial Intelligence*, 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 *Advances in Neural Information Processing Systems*, 2002.

J. Lee, M. Hwang, and J. J. Whang. PAC-Bayesian generalization bounds for knowledge graph representation learning. In *International Conference on Machine Learning*, 2024.

R. Levie. A graphon-signal analysis of graph neural networks. In *Advances in Neural Information Processing Systems*, 2023.

R. Levie, F. Monti, X. Bresson, and M. M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2019.

P. Li, Y. Wang, H. Wang, and J. Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. In *Advances in Neural Information Processing Systems*, 2020.

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

R. Liao, R. Urtasun, and R. S. Zemel. A PAC-Bayesian approach to generalization bounds for graph neural networks. In *International Conference on Machine Learning*, 2021.

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

S. Maskey, Y. Lee, R. Levie, and G. Kutyniok. Generalization analysis of message passing neural networks on large random graphs. In *Advances in Neural Information Processing Systems*, 2022.

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 *Annual Conference on Learning Theory*, 1999.

D. A. McAllester. Simplified pac-bayesian margin bounds. In *Annual Conference on Learning Theory*, 2003.

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.

A. Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

A. Micheli and A. S. Sestito. A new neural network model for contextual processing of graphs. In *Italian Workshop on Neural Nets Neural Nets and International Workshop on Natural and Artificial Immune Systems*, pages 10–17, 2005.

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

F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5425–5434, 2017.

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*, 2019.

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 *Advances in Neural Information Processing Systems*, 2020b.

C. Morris, F. Geerts, J. Tönshoff, and M. Grohe. WL meet VC. In *International Conference on Machine Learning*, 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.

L. Müller, M. Galkin, C. Morris, and L. Rampásek. Attending to graph transformers. *arXiv preprint*, 2023.

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

H. Pei, B. Wei, K. C. Chang, Y. Lei, and B. Yang. Geom-gcn: Geometric graph convolutional networks. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

P. Pellizzoni, T. Schulz, D. Chen, and K. M. Borgwardt. On the expressivity and sample complexity of node-individualized graph neural networks. In *Advances in Neural Information Processing Systems*, 2024.

C. Qian, D. Chételat, and C. Morris. Exploring the power of graph neural networks in solving linear optimization problems. *arXiv preprint*, abs/2310.10603, 2023.

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

A. Sadeghian, M. Armandpour, P. Ding, and D. Z. Wang. Drum: End-to-end differentiable rule mining on knowledge graphs. *Advances in Neural Information Processing Systems*, 32, 2019.

V. G. Satorras, E. Hoogeboom, and M. Welling. E(n) equivariant graph neural networks. In *International Conference on Machine Learning*, 2023.

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.

L. Scavuzzo, K. I. Aardal, A. Lodi, and N. Yorke-Smith. Machine learning augmented branch and bound for mixed integer linear programming. *arXiv preprint*, 2024.

M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *The semantic web: 15th International Conference*, 2018a.

M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web - International Conference*, 2018b.

P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29(3):93–106, 2008.

A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–35, 1997.

B. Srinivasan and B. Ribeiro. On the equivalence between positional node embeddings and structural graph representations. In *International Conference on Learning Representations*, 2020.

H. Tang and Y. Liu. Towards understanding generalization of graph neural networks. In *International Conference on Machine Learning*, 2023.

K. K. Teru, E. G. Denis, and W. L. Hamilton. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, 2020.

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

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

S. Vashishth, S. Sanyal, V. Nitin, and P. Talukdar. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*, 2020a.

S. Vashishth, S. Sanyal, V. Nitin, and P. P. Talukdar. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*, 2020b.

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

A. Vasileiou, S. Jegelka, R. Levie, and C. Morris. Survey on generalization theory for graph neural networks. *arXiv preprint*, 2024b.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

S. Verma and Z. Zhang. Stability and generalization of graph convolutional neural networks. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2019.

X. Wang, H. Yang, and M. Zhang. Neural common neighbor with completion for link prediction. In *International Conference on Learning Representations*, 2024a.

X. Wang, H. Yang, and M. Zhang. Neural common neighbor with completion for link prediction. In *International Conference on Learning Representations*, 2024b.

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.

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

K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

K. Xu, M. Zhang, S. Jegelka, and K. Kawaguchi. Optimization of graph neural networks: Implicit acceleration by skip connections and more depth. In *International Conference on Learning Representations*, 2021.

F. Yang, Z. Yang, and W. W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. *Advances in Neural Information Processing Systems*, 30, 2017.

Z. Yang, W. W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In M. Balcan and K. Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, JMLR Workshop and Conference Proceedings, 2016.

Z. Ye, Y. J. Kumar, G. O. Sing, F. Song, and J. Wang. A comprehensive survey of graph neural networks for knowledge graphs. *IEEE Access*, 10:75729–75741, 2022.

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

J. You, J. Gomes-Selman, R. Ying, and J. Leskovec. Identity-aware graph neural networks. In *AAAI Conference on Artificial Intelligence*, pages 10737–10745, 2021.

D. Yu, Y. Yang, R. Zhang, and Y. Wu. Generalized multi-relational graph convolution network. In *ICML Workshop on Graph Representation Learning and Beyond*, 2020.

S. Yun, S. Kim, J. Lee, J. Kang, and H. J. Kim. Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction. In *Advances in Neural Information Processing Systems*, 2021.

M. Zhang and Y. Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, 2018.

M. Zhang, P. Li, Y. Xia, K. Wang, and L. Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. In *Advances in Neural Information Processing Systems*, 2021.

X. Zhang, L. Wang, J. Helwig, Y. Luo, C. Fu, Y. Xie, M. Liu, Y. Lin, Z. Xu, K. Yan, K. Adams, M. Weiler, X. Li, T. Fu, Y. Wang, H. Yu, Y. Xie, X. Fu, A. Strasser, S. Xu, Y. Liu, Y. Du, A. Saxton, H. Ling, H. Lawrence, H. Stärk, S. Gui, C. Edwards, N. Gao, A. Ladera, T. Wu, E. F. Hofgard, A. M. Tehrani, R. Wang, A. Daigavane, M. Bohde, J. Kurtin, Q. Huang, T. Phung, M. Xu, C. K. Joshi, S. V. Mathis, K. Azizzadenesheli, A. Fang, A. Aspuru-Guzik, E. J. Bekkers, M. M. Bronstein, M. Zitnik, A. Anandkumar, S. Ermon, P. Liò, R. Yu, S. Günnemann, J. Leskovec, H. Ji, J. Sun, R. Barzilay, T. S. Jaakkola, C. W. Coley, X. Qian, X. Qian, T. E. Smidt, and S. Ji. Artificial intelligence for science in quantum, atomistic, and continuum systems. *arXiv preprint*, 2023.

Z. Zhu, Z. Zhang, L. A. C. Xhonneux, and J. Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. In *Advances in Neural Information Processing Systems*, 2021.

A Related works

In the following, we discuss relevant related work.

Graph neural networks GNNs are a family of neural architectures designed to compute vectorial representations for the nodes of a given graph while encoding structural information about both the graph and its nodes. Recently, MPNNs [Gilmer et al., 2017, Scarselli et al., 2009] have emerged as the most prominent architecture in graph machine learning. Notable instances of this architecture include, e.g., Duvenaud et al. [2015], Hamilton et al. [2017], Kipf and Welling [2017], and Veličković et al. [2018], which can be subsumed under the message-passing framework introduced in Gilmer et al. [2017]. In parallel, approaches based on spectral information were introduced in, e.g., Bruna et al. [2014], Defferrard et al. [2016], Gama et al. [2019], Kipf and Welling [2017], Levie et al. [2019], and Monti et al. [2017]—all of which descend from early work in Baskin et al. [1997], Goller and Küchler [1996], Kireev [1995], Merkwirth and Lengauer [2005], Micheli and Sestito [2005], Micheli [2009], Scarselli et al. [2009], and Sperduti and Starita [1997]. MPNNs have been applied to graph-, node-, and link-level prediction settings [Chami et al., 2020, Vasileiou et al., 2024a].

Node and link prediction using MPNNs Since MPNNs compute a vectorial representation for each node, utilizing them for node-level prediction is straightforward. However, using them for link prediction is less straightforward. Hence, an extensive set of papers proposing MPNN architectures for link prediction exists [Ye et al., 2022]. One of the earliest approaches, e.g., [Kipf and Welling, 2016, Schlichtkrull et al., 2018b, Vashishth et al., 2020b], used MPNNs to compute a vectorial representation for each node, which is subsequently used to predict the existence of a link between two nodes. However, such two-stage approaches are not expressive enough for link prediction [Srinivasan and Ribeiro, 2020, Zhang et al., 2021]. Zhang and Chen [2018] proposed an architecture that, for two given nodes v and w, computes the union of the subgraphs induced by all nodes within a pre-specified shortest-path distance of either v or w, labels nodes with their distances according to v and w, respectively, and uses an MPNN on top of this subgraph to predict the existence of a link between v and w. Teru et al. [2020] proposed a related method, taking the intersection instead of the union. Zhang et al. [2021], introduced the *labeling trick* for MPNNs, which, assuming a directed graph, essentially labels the "source" and "target" nodes with unique labels, and potentially labels the other nodes with specific labels as well, and runs an MPNN on top of this specifically labeled graph to predict the existence of a link between the two nodes. They showed that their labeling trick framework encompasses other architectures, such as Li et al. [2020], You et al. [2021]. A more refined version of this idea, also leading to improved empirical results, was introduced in Zhu et al. [2021], introducing *NBFNet*. See also Kong et al. [2022], Chamberlain et al. [2023], Wang et al. [2024a], Yun et al. [2021] for more efficient variants. Subsequently, the works of Zhang et al. [2021], Zhu et al. [2021] were studied theoretically in Huang et al. [2023], showing that their expressive power is upper bounded by a local variant of the 2-dimensional Weisfeiler–Leman algorithm [Morris et al., 2020b, Barceló et al., 2022] and introduced an MPNN with the same expressive power as the former.

An extensive set of works proposes MPNN architectures for link prediction in knowledge graphs, including embedding and path-based approaches predating MPNN approaches; see Ali et al. [2022], Ye et al. [2022] for surveys.

Generalization analysis of MPNNs for node-level prediction In a first attempt to understand the generalization abilities of MPNNs, Scarselli et al. [2018] leveraged classical techniques from learning theory [Karpinski and Macintyre, 1997] to show that MPNNs' *VC dimension* [Vapnik, 1995] for node-level prediction tasks with piece-wise polynomial activation functions on a *fixed* graph, under various assumptions, is in $\mathcal{O}(P^2 n \log n)$, where P is the number of parameters and n is the order of the input graph; see also Hammer [2001]. We note here that Scarselli et al. [2018] analyzed a different type of MPNN not aligned with modern MPNN architectures [Gilmer et al., 2017]; see also D'Inverno et al. [2024]. Moreover, the work does not account for non-i.i.d. samples, and due to their reliance on VC dimension theory, they are bound to the binary-classification task and the use of the impractical 0-1 loss function. Verma and Zhang [2019] derived generalization bounds for node classification tasks, assuming that nodes are sampled in an i.i.d. fashion from the given graph. They utilized the algorithmic stability [Bousquet and Elisseeff, 2002] to derive a generalization error bound for a single-layer MPNN layer, demonstrating that the *algorithmic stability* property strongly depends on the largest absolute eigenvalue of the graph convolution filter.

In the transductive setting, building on the *transductive Rademacher average framework* of El-Yaniv and Pechyony [2007], Esser et al. [2021] derived generalization bounds that depend on the maximum norm (maximum absolute row sum) of the graph operator, e.g., the adjacency matrix or the graph's Laplacian; see also Tang and Liu [2023] for refined results. Baranwal et al. [2021] studied the classification of a mixture of Gaussians, where the data corresponds to the node features of a stochastic block model, deriving conditions under which the mixture model is linearly separable using the GCN layer [Kipf and Welling, 2017].

Generalization analysis of MPNNs for link-level prediction There is little work analyzing link prediction architectures' generalization abilities. Lee et al. [2024] analyzed a large set of transductive link prediction architectures for knowledge graphs using a PAC-Bayesian analysis [McAllester, 1999, 2003, Langford and Shawe-Taylor, 2002]. However, Lee et al. [2024] are restricted to the impractical margin loss, only consider less expressive MPNN-based link prediction architectures, e.g., Schlichtkrull et al. [2018b], Vashishth et al. [2020b], not considering modern link prediction architectures, and do not account for the influence of graph structure.

Generalization analysis of MPNNs for graph-level prediction Garg et al. [2020] showed that the empirical Rademacher complexity (see, e.g., Mohri et al. [2018]) of a specific, simple MPNN architecture, using sum aggregation and specific margin loss, is bounded in the maximum degree, the number of layers, Lipschitz constants of activation functions, and parameter matrices' norms. We note here that their analysis assumes weight sharing across layers. Recently, Karczewski et al. [2024] lifted this approach to $E(n)$-equivariant MPNNs [Satorras et al., 2023]. Liao et al. [2021] refined the results of Garg et al. [2020] via a PAC-Bayesian approach, further refined in Ju et al. [2023]. Maskey et al. [2022, 2024] assumed that data is generated by random graph models, leading to MPNNs' generalization analysis depending on the (average) number of nodes of the graphs. In addition, Levie [2023] and Rauchwerger et al. [2024] defined metrics on attributed graphs, resulting in a generalization bound for MPNNs depending on the covering number of these metrics. Recently, Morris et al. [2023] made progress connecting MPNNs' expressive power and generalization ability via the Weisfeiler–Leman hierarchy. They studied the influence of graph structure and the parameters' encoding lengths on MPNNs' generalization by tightly connecting 1-*dimensional Weisfeiler–Leman algorithm* (1-WL) expressivity and MPNNs' VC dimension. They derived that MPNNs' VC dimension depends tightly on the number of equivalence classes computed by the 1-WL over a given set of graphs. Moreover, they showed that MPNNs' VC dimension depends logarithmically on the number of colors computed by the 1-WL and polynomially on the number of parameters. Since relying on the 1-WL, their analysis implicitly assumes a discrete pseudo-metric space, where two graphs are either equal or far apart. One VC lower bound reported in Morris et al. [2023] was tightened in Daniëls and Geerts [2024] to MPNNs restricted to using a single layer and a width of one. In addition, Pellizzoni et al. [2024] extended the analysis of Morris et al. [2023] to node-individualized MPNNs and devised a Rademacher-complexity-based approach using a covering number argument Bartlett et al. [2017]. Franks et al. [2023] studied MPNNs' VC dimension assuming linearly separable data and showed a tight relationship to the data's margin, also partially explaining when more expressive architectures lead to better generalization. Li et al. [2024] 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. Kriege et al. [2018] leveraged 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]. Most recently, building on the robustness framework of Xu and Mannor [2012], Vasileiou et al. [2024a] derived MPNNs' generalization abilities for graph-level predictions by studying different pseudo-metrics capturing MPNNs' computation, improving over the results in Morris et al. [2023]. Finally, Yehudai et al. [2021] showed negative results for MPNNs' generalization ability to larger graphs.

See Vasileiou et al. [2024b] for a survey on generalization analyses of MPNNs and related architectures.

B Extended background

B.1 Detailed notation

The following summarizes our notation in detail.

Basic notations Let $\mathbb{N} \coloneqq \{1, 2, \ldots\}$ 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 $[n] \coloneqq \{1, \ldots, n\} \subset \mathbb{N}$. We use $\{\!\!\{\ldots\}\!\!\}$ to denote multisets, i.e., the generalization of sets allowing for multiple, finitely many instances for each of its elements. For an arbitrary set X, we denote by 2^X the set consisting of all possible subsets of X. For two non-empty sets

X and Y, let Y^X denote the set of functions from X to Y. Given a set X and a subset $A \subset X$, we define the indicator function $1_A : X \to \{0, 1\}$ such that $1_A(x) = 1$ if $x \in A$, and $1_A(x) = 0$ otherwise. 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 $\boldsymbol{M}$. Let $\boldsymbol{N}$ be an $n \times n$ matrix, $n > 0$, then the *trace* $\operatorname{Tr}(\boldsymbol{N}) := \sum_{i \in [n]} N_{ii}$. In what follows, $\mathbf{o}$ denotes an all-zero vector with an appropriate number of components.

Norms Given a vector space V, a *norm* is a function $\| \cdot \| : V \to \mathbb{R}^+$ which satisfies the following properties. For all vectors $\boldsymbol{u}, \boldsymbol{v} \in V$ and scalar $s \in \mathbb{R}$, we have (i) *non-negativity,* $\|\boldsymbol{v}\| \geq 0$ with $\|\boldsymbol{v}\| = 0$ if, and only, if $\boldsymbol{v} = \mathbf{o}$; (ii) *scalar multiplication,* $\|s\boldsymbol{v}\| = |s| \, \|\boldsymbol{v}\|$; and the (iii) *triangle inequality* holds, $\|\boldsymbol{u} + \boldsymbol{v}\| \leq \|\boldsymbol{u}\| + \|\boldsymbol{v}\|$. When V is some real vector space, say $\mathbb{R}^{1 \times d}$, for $d > 0$, here, and in the remainder of the paper, $\| \cdot \|_1$ and $\| \cdot \|_2$ refer to the 1-*norm* $\|\boldsymbol{x}\|_1 := |x_1| + \cdots + |x_d|$ and 2-*norm* $\|\boldsymbol{x}\|_2 := \sqrt{x_1^2 + \cdots + x_d^2}$, respectively, for $\boldsymbol{x} \in \mathbb{R}^{1 \times d}$. When considering the vector space $\mathbb{R}^{n \times n}$ of square $n \times n$ matrices, a *matrix norm* $\| \cdot \|$ is a norm as described above, with the additional property that $\|\boldsymbol{M}\boldsymbol{N}\| \leq \|\boldsymbol{M}\| \|\boldsymbol{N}\|$ for all matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ in $\mathbb{R}^{n \times n}$. Finally, for two vectors $\boldsymbol{u} \in \mathbb{R}^{d_1}$ and $\boldsymbol{v} \in \mathbb{R}^{d_2}$, we denote by $\boldsymbol{u} \| \boldsymbol{v} \in \mathbb{R}^{d_1 + d_2}$ the concatenation of the two vectors.

Graphs An *(undirected) graph* G is a pair $(V(G), E(G))$ with *finite* sets of *nodes* $V(G)$ and *edges* $E(G) \subseteq \{\{u, v\} \subseteq V(G) \mid u \neq v\}$. The *order* of a graph G is its number $|V(G)|$ of nodes. We denote the set of all n-order (undirected) graphs by $\mathcal{G}_n$. In a *directed graph*, we define $E(G) \subseteq V(G)^2$, where each edge (u, v) has a direction from u to v. The *chromatic number* $\chi(G)$ of a graph G is the minimum number of colors required to color the nodes of G such that no two adjacent nodes share the same color. Given a directed graph G and nodes $u, v \in V(G)$, we say that v is a *child* of u if $(u, v) \in E(G)$. If a node has no children, we refer to this node as a *leaf*. Given a (directed) graph G and nodes $u, v \in V(G)$, we call a path from u to v a tuple $p = (u_1, \ldots, u_{k+1}) \in V(G)^k$ such that $(u_i, u_{i+1}) \in E(G)$, for all $i \in [k]$, $u_i \neq u_j$, for $i \neq j$, $u_1 = u$ and $u_{k+1} = v$. We refer to k as the length of the path, and we write length$(p) = k$. We denote by $\mathcal{P}_G(u, v)$ the set of all possible paths from u to v on a graph G, and by $\mathcal{P}^{(k)}(u, v)$ the set of all possible paths from u to v with length k on a graph G. Analogously, for undirected graphs by replacing (u_{i-1}, u_i) with $\{u_{i-1}, u_i\}$. A graph G is called *connected* if, for any $u, v \in V(G)$, a path exists from u to v. We say that a graph G is *disconnected* if it is not connected. For an n-order graph $G \in \mathcal{G}_n$, assuming $V(G) = [n]$, 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 $\{v, w\} \in E(G)$. For a graph G, the k-*neighborhood* of a node $v \in V(G)$ denoted by $N_G^{(k)}(v)$ contains all nodes in a path of length at most k from v. When $k = 1$, we refer to elements of $N_G^{(1)}(v)$ as neighbors of v and we omit k in the notation. The *degree* of a node v is $|N_G(v)|$. When referring to a directed graph, we use the notation $N_{G,\text{out}}^{(k)}(v)$. For $S \subseteq V(G)$, the graph $G[S] := (S, E_S)$ is the *subgraph induced by* S, where $E_S := \{(u, v) \in E(G) \mid u, v \in S\}$. A *(node-)featured graph* is a pair (G, a_G) with a graph $G = (V(G), E(G))$ and a function $a_G : V(G) \to \Sigma$, where Σ is an arbitrary set. Similarly, we define *(edge-)featured graphs*. For a node $v \in V(G)$, $a_G(v)$ denotes its *feature*. We denote the class of all (undirected) graphs with d-dimensional, real-valued node features by $\mathcal{G}_d^{\mathbb{R}}$. A *knowledge graph* is a directed edge-featured graph $G = (V(G), E(G), R)$, where $E(G) \subseteq V \times V \times R$ for a finite set R of relation types. Each edge $(v, x, r) \in E(G)$ is labeled with a relation type $r \in R$. For a node $v \in V$, we define its *relation-aware neighborhood* as $N_r(v) := \{(x, r) \in V \times R \mid (v, x, r) \in E(G)\}$, where each neighbor x is connected to v by an edge labeled with relation r. A *query vector* $\boldsymbol{w}_q \in \mathbb{R}^d$ encodes a query-specific signal that influences how information is passed along edges during message propagation. For each edge (v, x, r), the embedding $\boldsymbol{w}_q(v, x, r)$ helps the model focus on paths and structures that are relevant to the query q.

Trees A graph G is a *tree* if it is connected, but for any $e \in E(G)$ the graph $G \setminus \{e\}$ with $V(G \setminus \{e\}) = V(G)$ and $E(G \setminus \{e\}) = E(G) \setminus \{e\}$ is disconnected. A tree or a disjoint collection of trees is known as a *forest*. A *rooted tree* (T, r) or T_r is a tree where a specific node $r \in V(T)$ is marked as the *root*. For a rooted (undirected) tree T_r, we can define an implicit direction on all edges as pointing away from the root; thus, when we refer to the *children* of a node u in a rooted tree, we implicitly consider this directed structure. To distinguish between a rooted tree T_r and its directed counterpart, we use the notation $\overrightarrow{T_r}$ to denote directed rooted trees. For a rooted tree (T, r), and $L \in \mathbb{N}$, we define the L level of (T, r) as the set of nodes $v \in V(T)$ satisfying the equation $\min\{\text{length}(p) \mid p \in \mathcal{P}_T(r, u)\} = L$.

Graph isomorphisms 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 node (or edge)-featured graphs, we additionally require that $a_G(v) = a_H(\varphi(v))$ for $v \in V(G)$ (or $a_G(e) = a_H(\varphi(e))$ for $e \in E(G)$) and for rooted trees, we further require that the root is mapped to the root. Moreover, we call the equivalence classes induced by $\simeq$ *isomorphism types* and denote the isomorphism type of G by $\tau(G)$. A *graph class* is a set of graphs closed under isomorphism.

B.2 Metric spaces and continuity

The following summarizes the foundational concepts of metric spaces and Lipschitz continuity used throughout this work.

Metric spaces In the remainder of the paper, "distances" between graphs play an essential role, which we make precise by defining a *pseudo-metric* (on the set of graphs). Let $\mathcal{X}$ be a set equipped with a pseudo-metric $d\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$, i.e., d is a function satisfying $d(x, x) = 0$ and $d(x, y) = d(y, x)$ for $x, y \in \mathcal{X}$, and $d(x, y) \leq d(x, z) + d(z, y)$, for $x, y, z \in \mathcal{X}$. The latter property is called the triangle inequality. The pair $(\mathcal{X}, d)$ is called a *pseudo-metric space*. For $(\mathcal{X}, d)$ to be a *metric space*, d additionally needs to satisfy $d(x, y) = 0 \Rightarrow x = y$, for $x, y \in \mathcal{X}$.[1]

Covering numbers and partitions Let $(\mathcal{X}, d)$ be a pseudo-metric space. Given an $\varepsilon > 0$, an *ε-cover* of $\mathcal{X}$ is a subset $C \subseteq \mathcal{X}$ such that for all elements $x \in \mathcal{X}$ there is an element $y \in C$ such that $d(x, y) \leq \varepsilon$. Given $\varepsilon > 0$ and a pseudo-metric d on the set $\mathcal{X}$, we define the *covering number* of $\mathcal{X}$,

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

i.e., the smallest number m such that there exists a ε-cover of cardinality m of the set $\mathcal{X}$ with regard to the pseudo-metric d.

The covering number provides a direct way of constructing a partition of $\mathcal{X}$. Let $K \coloneqq \mathcal{N}(\mathcal{X}, d, \varepsilon)$ so that, by definition of the covering number, there is a subset $\{x_1, \dots, x_K\} \subset \mathcal{X}$ representing an ε-cover of $\mathcal{X}$. We define a partition $\{C_1, \dots, C_K\}$ where

$$C_i \coloneqq \{x \in \mathcal{X} \mid d(x, x_i) = \min_{j \in [K]} d(x, x_j)\},$$

for $i \in [K]$. To break ties, we take the smallest i in the above. Observe that $\mathcal{X} = \bigcup_{i \in [K]} C_i$. We recall that the *diameter* of a set is the maximal distance between any two elements in the set. Implied by the definition of an ε-cover and the triangle inequality, each C_i has a diameter of at most 2ε.

Continuity on metric spaces Let $(\mathcal{X}, d_{\mathcal{X}})$ and $(\mathcal{Y}, d_{\mathcal{Y}})$ be two pseudo-metric spaces. A function $f\colon \mathcal{X} \to \mathcal{Y}$ is called *c_f-Lipschitz continuous* if, for $x, x' \in \mathcal{X}$,

$$d_{\mathcal{Y}}(f(x), f(x')) \leq c_f \cdot d_{\mathcal{X}}(x, x').$$

C Measure Theory and Random Variables

In this section, we provide the necessary background from measure theory. Specifically, we formally define measure spaces, probability spaces, random variables, distributions, and expectations.

Definition 5 (σ-algebra)**.** Let X be a set, and let $2^X = \{A \mid A \subset X\}$ denote its power set. A subset $\mathcal{F} \subset 2^X$ is called a *σ-algebra* on X if it satisfies the following properties:

- $X \in \mathcal{F}$,
- If $A \in \mathcal{F}$, then $X \setminus A \in \mathcal{F}$ (closed under complementation),
- If $\{A_n\}_{n \in \mathbb{N}}$ is a sequence of sets in $\mathcal{F}$, then $\bigcup_{n \in \mathbb{N}} A_n \in \mathcal{F}$ (closed under countable unions).

If X is a set and $\mathcal{F}$ is a σ-algebra on X, the pair $(X, \mathcal{F})$ is called a *measurable space*.

[1]Observe that computing a metric on the set of graphs $\mathcal{G}$ up to isomorphism is at least as hard as solving the graph isomorphism problem on $\mathcal{G}$.

Note that for any set X, the power set 2^X and the set $\{\emptyset, X\}$ are σ-algebras on X. Furthermore, given a set X and a collection $\mathcal{F} \subset 2^X$, we define $\sigma(\mathcal{F})$ as the smallest σ-algebra on X containing $\mathcal{F}$. This is well defined, as 2^X is always a σ-algebra. Verifying that a countable union of σ-algebras is again a σ-algebra is also straightforward.

Definition 6 (Measure space)**.** Given a measurable space $(X, \mathcal{F})$, a function $\mu\colon \mathcal{F} \to \mathbb{R}^+$ is called a *measure* on $(X, \mathcal{F})$ if:

- $\mu(\emptyset) = 0$,
- If $\{A_n\}_{n\in\mathbb{N}}$ is a collection of pairwise disjoint sets in $\mathcal{F}$, then
$$\mu\left(\bigcup_{n\in\mathbb{N}} A_n\right) = \sum_{n\in\mathbb{N}} \mu(A_n).$$

The triplet $(X, \mathcal{F}, \mu)$ is called a *measure space*.

If, in addition, $\mu(X) = 1$, then $(X, \mathcal{F}, \mu)$ is called a *probability space*, and μ is referred to as a *probability measure*. Below, we formally define random variables and their distributions.

Definition 7 (Random variable and distribution)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space and $(E, \mathcal{E})$ a measurable space. A function $X : \Omega \to E$ is called a *random variable* if it is $\mathcal{E}$-measurable, i.e.,
$$X^{-1}(A) \in \mathcal{F}, \quad \forall A \in \mathcal{E}.$$
Furthermore, the function $P^X : \mathcal{E} \to [0, 1]$ defined by
$$P^X(A) = P(\{\omega \mid X(\omega) \in A\})$$
is a probability measure on $(E, \mathcal{E})$, and it is called the *distribution* induced by X.

If two random variables X and Y satisfy $P^X \equiv P^Y$, then we say that X and Y are *identically distributed*. Additionally, when $E = \mathbb{R}^d$ for some $d \in \mathbb{N}$, the σ-algebra $\mathcal{E}$ is implicitly assumed to be the Borel σ-algebra, denoted by $\mathcal{B}(\mathbb{R}^d)$. This is the smallest σ-algebra containing all open subsets of $\mathbb{R}^d$. In any other case where the σ-algebra is omitted, we implicitly refer to the power set of the space.

Following the notation in the definition of a random variable, given a random variable X, we denote by $\sigma(X)$ the smallest σ-algebra on E such that $X\colon (\Omega, \mathcal{F}) \to (E, \sigma(X))$ is measurable. Below, we introduce the concept of dependence between random variables. Note that all integrals below refer to the Lebesgue integral.

Definition 8 (Expectation and conditional expectation)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space, and let $X\colon \Omega \to \mathbb{R}$ be a random variable. Given $A \in \mathcal{F}$ with $P(A) > 0$, we define:

- The expectation of X:
$$\mathbb{E}(X) = \int_\Omega X(\omega) P(d\omega).$$
If $\mathbb{E}(|X|) < \infty$, we say that X is *integrable*.
- The conditional expectation of X given event A:
$$\mathbb{E}(X|A) = \frac{1}{P(A)} \int_A X(\omega) P(d\omega).$$

In the above setting, if $\mathcal{F}' \subset \mathcal{F}$ is another σ-algebra on X and X is integrable, it can be shown that there exists a unique random variable Y satisfying (i) integrability, (ii) $\mathcal{F}'$-measurability, and (iii) for all $A \in \mathcal{F}'$, with $P(A) > 0$, we have $\mathbb{E}(X|A) := \mathbb{E}(Y|A)$. We denote this random variable as $\mathbb{E}[X|\mathcal{F}']$, which is called the *expectation of X conditioned on $\mathcal{F}'$*. Consequently, given two random variables X and Y, we define the conditional expectation of X given Y as $\mathbb{E}(X|Y) = \mathbb{E}(X|\sigma(Y))$.

Another important result concerning the expectation of random variables is the so-called change of measure formula. That is if $X\colon \Omega \to E$ is a random variable and $h\colon E \to \mathbb{R}$ is a measurable function, then
$$\int_\Omega h(X(\omega)) P(d\omega) = \int_E h(x) P^X(dx).$$
This formula expresses the expectation of $h(X)$ as an integral concerning the pushforward measure P^X, rather than the original probability measure P. Lastly, we state the *tower property*, useful in martingale theory.

Lemma 9 (Tower property)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space, and let $X \colon \Omega \to \mathbb{R}$ be an integrable random variable. If $\mathcal{F}_1, \mathcal{F}_2 \subset \mathcal{F}$ are two σ-algebras such that $\mathcal{F}_1 \subset \mathcal{F}_2$, then

$$\mathbb{E}\big(\mathbb{E}(X|\mathcal{F}_2) \mid \mathcal{F}_1\big) = \mathbb{E}(X \mid \mathcal{F}_1).$$

As a corollary, for any two random variables X and Y, we have the well-known formula:

$$\mathbb{E}(\mathbb{E}(X|Y)) = \mathbb{E}(X).$$

Note that using the definition of the expectation and the above properties we can derive many useful results in probability theory using the following simple observation $\mathbb{E}(\mathbf{1}_{\{X\in A\}}) = P(A)$.

Definition 10 (Independence)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space, and let $(E_i, \mathcal{E}_i)$ for $i = 1, 2$ be measurable spaces. Suppose $X_i \colon \Omega \to E_i$ are random variables for $i = 1, 2$. We say that X_1 and X_2 are *independent* (denoted as $X_1 \perp X_2$) if

$$P(X_1 \in A, X_2 \in B) = P^{X_1}(A)P^{X_2}(B), \quad \forall A \in \mathcal{E}_1, B \in \mathcal{E}_2.$$

Two sets of random variables S_1 and S_2 are independent if every pair (X_1, X_2) with $X_1 \in S_1$ and $X_2 \in S_2$ is independent.

D Martingales

This section introduces martingale theory, which is essential when dealing with non-independent data. We derive a useful concentration inequality, known as the Azuma inequality, which is analogous to Hoeffding's inequality. Recall that Hoeffding's inequality provides a probabilistic bound on the deviation of a sum of independent random variables from its expectation. Similarly, Azuma's inequality establishes a bound for this deviation in cases where the random variables are not independent but exhibit a controlled dependence.

Before formally defining a martingale, we introduce the notion of a filtration on a measurable space. Let $(\Omega, \mathcal{F}, P)$ be a probability space, and let $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$ be an increasing sequence of σ-algebras (i.e., $\mathcal{F}_n \subset \mathcal{F}_{n+1}$ for all $n \in \mathbb{N}$) such that $\mathcal{F}_n \subset \mathcal{F}$. Then, $\{\mathcal{F}_n \mid n \in \mathbb{N}\}$ is called a filtration on $(\Omega, \mathcal{F})$.

Definition 11. Let $(\Omega, \mathcal{F}, P)$ be a probability space equipped with a filtration $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$. A sequence of random variables $\{X_n\}_{n\in\mathbb{N}} \colon \Omega \to \mathbb{R}$ adapted to $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$, meaning that X_n is $\mathcal{F}_n$-measurable for all $n \in \mathbb{N}$, is called a martingale if:

(i) X_n is integrable for all $n \in \mathbb{N}$.

(ii) $\mathbb{E}(X_{n+1}|\mathcal{F}_n) = X_n$ for all $n \in \mathbb{N}$.

Below, we define the Doob martingale associated with a given random variable. The Doob martingale (or Lévy martingale) is a sequence of random variables approximating the given random variable while satisfying the martingale property concerning a given filtration.

Definition 12 (Doob martingale)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space, $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$ be a filtration, and $X \colon \Omega \to \mathbb{R}$ be an integrable random variable. We define the sequence $\{W_n\}_{n\in\mathbb{N}}$ inductively as follows:

$$W_0 = \mathbb{E}(X), \quad \text{and} \quad W_n = \mathbb{E}(X \mid \mathcal{F}_n), \text{ for } n \geq 1.$$

Using the tower property of conditional expectation, it is easy to verify that $\{W_n\}_{n\in\mathbb{N}}$ is a martingale concerning the filtration $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$. The sequence $\{W_n\}_{n\in\mathbb{N}}$ is called the Doob martingale of X with respect to the filtration $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$.

The usefulness of the above definition becomes evident when considering the case where X is a function of n random variables. That is, let $Y_1, \ldots, Y_n \colon \Omega \to E$ and $f \colon E^n \to \mathbb{R}$. Define $X := f(Y_1, \ldots, Y_n)$, and let $\{W_n\}$ be the Doob martingale associated with the filtration $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$, where $\mathcal{F}_k = \cup_{i=1}^k \sigma(Y_i)$ for $k \in [n]$, and $\mathcal{F}_k = \mathcal{F}_n$ for $k > n$.

Then, we have that $W_n = f(Y_1, \ldots, Y_n)$ and $W_0 = \mathbb{E}(f(Y_1, \ldots, Y_n))$. Therefore, bounding the difference $W_n - W_0$ leads to a concentration inequality. The following result, known as Azuma's inequality [Azuma, 1967], describes the exact conditions under which this difference can be controlled.

Theorem 13 (Azuma's Inequality)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space, and let $\{W_n\}_{n\in\mathbb{N}}$ be a martingale adapted to a filtration $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$. Suppose there exist constants $c_k \in \mathbb{R}$ for $k \in \mathbb{N}$ such that, with probability 1,

$$|W_k - W_{k-1}| \leq c_k.$$

Then, for all $N \in \mathbb{N}$ and every $t > 0$, we have

$$P(|W_N - W_0| \geq t) \leq 2 \exp\left(\frac{-t^2}{2\sum_{k=1}^{N} c_k^2}\right).$$

E Formal definitions of generalized MPNNs and unrolling distances

As discussed, in link prediction tasks, MPNNs are often applied to a transformed version of the initially given graph to encode specific structural information, e.g., using the labeling trick [Zhang et al., 2021]. To allow for studying MPNNs' generalization properties in a unified fashion for graph-, node-, and link-level prediction, we derive *generalized MPNNs*.

Intuitively, generalized MPNNs compute vector representations for target entities (such as nodes, links, or tuples of nodes) depending on the task. They proceed in three steps. First, the input graph is transformed via a transformation function, and a standard MPNN (as defined in Equation (1)) is applied to compute node embeddings on the transformed graph. Secondly, a selection function chooses a set of nodes from the transformed graph for each target entity. Thirdly, for each target entity, vector representations of the selected nodes (from step 2) are aggregated via a pooling function to yield the final representation of the target entity.

We first specify the type of vectorial representation the MPNN aims to compute, e.g., node, link, or graph vectorial representations. Formally, let $d \in \mathbb{N}$, and $\mathcal{G}_d^{\mathbb{R}}$ be the set of all graphs with node features in $\mathbb{R}^d$. For a subset $\mathcal{G}' \subseteq \mathcal{G}_d^{\mathbb{R}}$, we define a *graph-representation task* on $\mathcal{G}'$ as $\mathcal{G}' \otimes \mathfrak{R}$, where

$$\mathcal{G}' \otimes \mathfrak{R} \subseteq \{(G, S) \mid G \in \mathcal{G}', S \subseteq V(G)\}.$$

Intuitively, the set $\mathcal{G}' \otimes \mathfrak{R}$ consists of the subset of nodes for which we aim to compute vectorial representations. For example, if the goal is to build a model for node classification on graphs in $\mathcal{G}_d^{\mathbb{R}}$, we consider the *node-representation task*,

$$\mathcal{G}_d^{\mathbb{R}} \otimes V \coloneqq \{(G, \{u\}) \mid G \in \mathcal{G}_d^{\mathbb{R}}, u \in V(G)\},$$

and similarly for the *link-representation task*,

$$\mathcal{G}_d^{\mathbb{R}} \otimes E \coloneqq \{(G, e) \mid G \in \mathcal{G}_d^{\mathbb{R}}, e \in V(G)^2\}.$$

Generalized MPNNs Based on this, we now define *generalized MPNNs* that encompass all the above tasks. Let $\mathcal{G}' \subseteq \mathcal{G}_d^{\mathbb{R}}$, given a graph-representation task $\mathcal{G}' \otimes \mathfrak{R}$ and $L \in \mathbb{N}$, a generalized MPNN with L layers, denoted as a $(\mathcal{T}, \mathcal{V}, \Psi)$-$\mathsf{MPNN}(L)$, computes vector representations for elements in $\mathcal{G}' \otimes \mathfrak{R}$ by applying an MPNN on a transformed graph. A generalized MPNN consists of three main components.

1. A *transformation function* $\mathcal{T} \colon \mathcal{G}' \to \mathcal{G}$
2. A *selection function* $\mathcal{V}$ determines, for a pair $(G, S) \in \mathcal{G}' \otimes \mathfrak{R}$, which nodes A in the transformed graph $\mathcal{T}(G)$ are used in the MPNN computation for (G, S). That is,
$$\mathcal{V} \colon (G, S) \mapsto A \subset V(\mathcal{T}(G)).$$
3. A *pooling function* computes the final representations for the elements in $\mathcal{G}' \otimes \mathfrak{R}$. That is, we first use an MPNN for L layers for computing vectorial representations for the nodes in the transformed graph using the selection function $\mathcal{V}(G, S)$, leading to the set,
$$F(G, S) = \{(\boldsymbol{h}_{\mathcal{T}(G)}^{(L)}(v), v) \mid v \in \mathcal{V}(G, S)\}.$$
Subsequently, we use the *pooling function*,
$$\Psi \colon F(G, S) \mapsto \Psi(F(G, S)) \in \mathbb{R}^d, \quad \text{for } (G, S) \in \mathcal{G}' \otimes \mathfrak{R},$$
to aggregate these vectorial representations and compute the final representations for the elements in $\mathcal{G}' \otimes \mathfrak{R}$.

The final outcome is denoted by $\boldsymbol{h}_{\mathcal{T},\mathcal{V},\Psi}^{(L)}(G, S)$, i.e.,

$$\boldsymbol{h}_{\mathcal{T},\mathcal{V},\Psi}^{(L)}(G, S) = \Psi(F(G, S)), \quad \text{for } (G, S) \in \mathcal{G}' \otimes \mathfrak{R}.$$

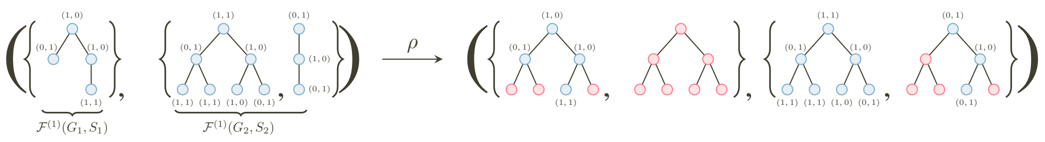
Figure 1: An illustration of the padding process described in Section 3. Red nodes indicate the padded (added) nodes with $\mathbf{o}_{\mathbb{R}^2}$ vertex features.

Unrolling distances In the following, we define distance functions that capture the computation performed by generalized MPNNs. Formally, let $\mathcal{G}' \subset \mathcal{G}_d^{\mathbb{R}}$ and let $(\mathcal{G}', \mathfrak{R})$ be a graph-representation task as previously defined. For technical reasons, we assume that for all $G \in \mathcal{G}'$ and all $u \in V(G)$, the node feature satisfies $a_G(u) \neq \mathbf{o}_{\mathbb{R}^d}$. The goal of this section is to define a family of distances on $(\mathcal{G}', \mathfrak{R})$ that are tailored to the structure of $(\mathcal{T}, \mathcal{V}, \Psi)$-MPNNs. These distances are parameterized by the transformation function $\mathcal{T}$ and the selection function $\mathcal{V}$, and are designed to satisfy the Lipschitz property, which is essential for our generalization analysis in the next section.

The definition of the distance is based on the notion of computation trees (also called unrolling trees), which are commonly used to describe the message-passing mechanism or characterize the 1-WL algorithm. For further background, see Appendix G. Given $(G, S) \in \mathcal{G}' \otimes \mathfrak{R}$, a transformation function $\mathcal{T}\colon \mathcal{G}' \to \mathcal{G}_d^{\mathbb{R}}$, a selection function $\mathcal{V}\colon (G, S) \mapsto A \subset V(\mathcal{T}(G))$, and a number of layers $L \in \mathbb{N}$, we define the multiset of unrolling trees as

$$\mathcal{F}^{(L)}(G, S) = \{\!\!\{ \mathsf{unr}(\mathcal{T}(G), u, L) \mid u \in \mathcal{V}(G, S) \}\!\!\}.$$

Following Vasileiou et al. [2024a], Chuang and Jegelka [2022], we aim to define a distance between two input pairs $(G_1, S_1), (G_2, S_2) \in \mathcal{G}' \otimes \mathfrak{R}$ that measures the misalignment between the multisets of unrolling trees $\mathcal{F}^{(L)}(G_1, S_1)$ and $\mathcal{F}^{(L)}(G_2, S_2)$. To handle multisets and unrollings of different sizes, we first define a padding function ρ, illustrated in Figure 1 and formally described in Appendix I.

We now define the unrolling distance on the domain $\mathcal{G}_d^{\mathbb{R}} \otimes \mathfrak{C} \coloneqq \{(G, S) \mid G \in \mathcal{G}_d^{\mathbb{R}},\ S \subseteq V(G)\}$.

Definition 14 (Unrolling Distance)**.** Let $\mathcal{T}\colon \mathcal{G}' \to \mathcal{G}_d^{\mathbb{R}}$ be a transformation function and $\mathcal{V}\colon (G, S) \mapsto A \subset V(\mathcal{T}(G))$ a selection function. For any $(G, S) \in \mathcal{G}' \otimes \mathfrak{C}$, the $(\mathcal{T}, \mathcal{V})$-*Unrolling Distance* of depth L, denoted by $\mathrm{UD}^{(L)}_{\mathcal{T},\mathcal{V}}$, is defined on $\mathcal{G}_d^{\mathbb{R}} \otimes \mathfrak{C}$ as follows. Let $(G_1, S_1), (G_2, S_2) \in \mathcal{G}' \otimes \mathfrak{C}$ and let $\rho(\mathcal{F}^{(L)}(G_1, S_1), \mathcal{F}^{(L)}(G_2, S_2)) = (M_1, M_2)$. Then,

$$\mathrm{UD}^{(L)}_{\mathcal{T},\mathcal{V}}((G_1, S_1), (G_2, S_2)) = \min_{\varphi \in S(F_1, F_2)} \sum_{x \in V(F_1)} \|a_{F_1}(x) - a_{F_2}(\varphi(x))\|_2, \tag{2}$$

where F_i is the forest formed by the disjoint union of all trees in M_i for $i = 1, 2$, and

$$S(G, H) \coloneqq \{\varphi\colon V(G) \to V(H) \mid \varphi \text{ is an edge-preserving bijection}\}.$$

Note that an edge-preserving bijection preserves the graph structure but not necessarily the node features and that for a feature graph G, we use $a_G(u)$ to denote the feature of node $u \in V(G)$.

In Appendix I, we show that the unrolling distance $\mathrm{UD}^{(L)}_{\mathcal{T},\mathcal{V}}$ is a well-defined pseudo-metric on $\mathcal{G}_d^{\mathbb{R}} \otimes \mathfrak{C}$ (see Lemma 19). We also introduce a weighted extension of this distance (see Equation (8)) and prove that if the pooling function of a generalized MPNN satisfies the sub-sum property, then it is Lipschitz continuous concerning its corresponding unrolling distance (see Proposition 20).

F Link prediction MPNNs

Here, we provide an overview of state-of-the-art MPNN-based link prediction architectures that form the basis of our analysis. While the architectures are presented in the general setting of knowledge graphs, for simplicity, our analysis focuses on homogeneous graphs (i.e., $|R| = 1$), considering only initial node features and ignoring edge features. Throughout this work, we treat all of these architectures within the framework of generalized MPNNs defined in Section 3, but we present their original formulations below for completeness.

SEAL and subgraph Information Zhang and Chen [2018] and Teru et al. [2020] propose to use subgraph information around the target nodes to compute more expressive architectures. Importantly, these methods allow for inductive sampling from training graphs, as they only require enclosed subgraphs. Formally, let (G, a_G) be an undirected attributed graph, let $u, v \in V(G)$ be two target nodes, and let $k > 0$. The SEAL architecture Zhang and Chen [2018] first extracts the subgraph S induced by all nodes with a shortest-path distance of most k from one of the target nodes. We label the nodes in S using a *double-radius node labeling* $d\colon V(S) \to \mathbb{R}^2$, where each node $w \in V(S)$ is labeled with its respective distance to the two target nodes u and v. For nodes with $d(w, x) = \infty$ or $d(w, y) = \infty$, the corresponding node label is set to 0. We then run an MPNN on top of the resulting node-labeled graph to compute a vectorial representation for the two target nodes. The idea was later generalized to the *labeling trick* [Zhang et al., 2021], allowing for arbitrary node and edge labeling and more expressive architectures.

Conditional message-passing neural networks To study the expressive power of link prediction architectures, Huang et al. [2023] proposed conditional message-passing neural networks (C-MPNNs), which cover architectures such as NBFNet [Zhu et al., 2021], NeuralLP [Yang et al., 2017], and DRUM [Sadeghian et al., 2019]. Given a (knowledge) graph G, C-MPNNs compute pairwise node representations in the graph G for a fixed query vector q and source node $u \in V(G)$. The computation of all pairs (u, v) is conditioned on the source node u to provide a node representation of v conditioned by u. Hence, a conditional vectorial node representation $\boldsymbol{h}_{v|u,q}^{(t+1)} \in \mathbb{R}^d$ is computed for L layers by using the conditional message-passing architecture,

$$\boldsymbol{h}_{v|u,q}^{(t+1)} := \mathsf{UPD}\Big(\boldsymbol{h}_{v|u,q}^{f(t)}, \mathsf{AGG}\big(\{\!\!\{\mathsf{MSG}_r(\boldsymbol{h}_{x|u,q}^{(t)}, \boldsymbol{w}_q) \mid x \in N_r(v), r \in R\}\!\!\}\big)\Big), \tag{3}$$

where $\boldsymbol{h}_{v|u,q}^{0} = \mathsf{INIT}(u, v, q)$ denotes an initialization function satisfying node distinguishability; i.e., $\mathsf{INIT}(u, v) \neq \mathsf{INIT}(u, u)$ for $u \neq v$. Moreover, the function $f\colon \mathbb{N} \to \mathbb{N}$ is usually set to the identity function. By setting $f(t) = 0$ in Equation (3), one obtains the Neural Bellman–Ford Networks (NBFNets), as defined by Zhu et al. [2021], as a special case.

Neural common neighbors Since SEAL, C-MPNN and other subgraph-based architectures for link prediction propose labeling a graph and then applying an MPNN to evaluate node representations, leading to bad scalability, Wang et al. [2024b] proposed to reverse this process. For this, they devise the *neural common neighbors framework* (NCN), offering an approach of using an MPNN architecture on the original graph and enhancing the subsequent pooling process using structural information. For two nodes i, j, the node representation is computed using a two-step process. In the first step, an L-layer MPNN computes vectorial representations $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ for the nodes i and j. In contrast to SEAL and other subgraph-based architectures, the MPNN computation is now concatenated with the node representations from the common k-hop neighborhoods of the two nodes to compute the target tuple representation $\boldsymbol{z}_{ij}$. Furthermore, the node representations are aggregated using the Hadamard product $\odot$,

$$\boldsymbol{z}_{ij} := \boldsymbol{h}_i \odot \boldsymbol{h}_j \Big\| \sum_{u \in N_G^k(i) \cap N_G^k(j)} \boldsymbol{h}_u.$$

This tuple representation is calculated for each proposed link and evaluated using a feed-forward neural network, resulting in representations of each target link tuple (i, j).

Feed-forward neural networks An L-layer *feed-forward neural network* (FNN), for $L \in \mathbb{N}$, is a parametric function $\mathsf{FNN}_{\boldsymbol{\theta}}^{(L)}\colon \mathbb{R}^{1\times d} \to \mathbb{R}$, $d > 0$, where $\boldsymbol{\theta} := (\boldsymbol{W}^{(1)}, \dots, \boldsymbol{W}^{(d)}) \subseteq \Theta$ and $\boldsymbol{W}^{(i)} \in \mathbb{R}^{d\times d}$, for $i \in [L-1]$, and $\boldsymbol{W}^{(L)} \in \mathbb{R}^{d\times 1}$, where

$$\boldsymbol{x} \mapsto \sigma\Big(\cdots\sigma\Big(\sigma\Big(\boldsymbol{x}\boldsymbol{W}^{(1)}\Big)\boldsymbol{W}^{(2)}\Big)\cdots\boldsymbol{W}^{(L)}\Big) \in \mathbb{R},$$

for $\boldsymbol{x} \in \mathbb{R}^{1\times d}$. Here, the function $\sigma\colon \mathbb{R} \to \mathbb{R}$ is an *activation function*, applied component-wisely, e.g., a *rectified linear unit* (ReLU), where $\sigma(x) := \max(0, x)$. For an FNN where we do not need to specify the number of layers, we write $\mathsf{FNN}_{\boldsymbol{\theta}}$.

F.1 Link prediction as generalized MPNNs

Below, we describe how the link prediction MPNN architectures discussed above, i.e., SEAL, C-MPNN, and NCN, can be viewed as special cases of the generalized MPNN framework introduced in Section 3.

SEAL Verifying that SEAL constitutes a special case of generalized MPNNs as defined above is straightforward. To establish this, we must appropriately define the transformation function $\mathcal{T}$, the selection function $\mathcal{V}$, and the pooling function Ψ. For simplicity, we assume a directed graph G without initial vertex features. For each pair $(u, v) \in V(G)^2$, with $u \neq v$, we define the node set $V_{uv} \coloneqq N_G(u) \cup N_G(v) \cup \{u, v\}$. We then compute the induced subgraph $\bar{G}_{uv} = G[V_{uv}]$. Vertex features are added to $\bar{G}_{uv}$ using the double-radius node labeling described in Zhang and Chen [2018], yielding the graph G_{uv}. We define the transformation and selection functions as follows,

$$\begin{aligned}&\mathcal{T}(G) = \dot{\cup}_{(u,v)\in V(G)^2} G_{uv}, \text{ for } u \neq v, \quad \text{and,}\\ &\mathcal{V}(G, (u, v)) = V(G_{uv}),\end{aligned}$$

where $\dot{\cup}_{(u,v)\in V(G)^2} G_{uv}$ denotes the disjoint union of graphs. Finally, we use sum-pooling as our pooling function Ψ.

C-MPNNs We show that C-MPNNs constitute a special case of generalized MPNNs. For simplicity, we again assume a graph G without vertex features and set $f(t) = 1$ in Equation (3). Additionally, we assume that the graph contains only one type of edge, i.e., $|R| = 1$. We construct a new graph G' with vertex set $V(G') = V(G)^2$, and define initial vertex features using the INIT function such that $\text{INIT}(u, u) \neq \text{INIT}(u, v)$ for all $u \neq v$. The following condition defines the edge relation in G',

$$\{(u, v), (x, y)\} \in E(G') \iff u = x \text{ and } y \in N_G(v).$$

The transformation and selection functions are then defined as:

$$\begin{aligned}&\mathcal{T}(G) = G', \quad \text{and}\\ &\mathcal{V}(G, (u, v)) = \{(u, w) \mid w \in N_G(v)\}.\end{aligned}$$

Again, we use sum-pooling as our pooling function Ψ. By the definition of C-MPNNs, we can slightly modify the transformation and selection function to capture the NBFNets.

Neural common neighbors For the neural common neighbors architecture, we have $\mathcal{G}' \otimes \mathfrak{R} = \{(G, E(G)) \mid G \in \mathcal{G}_d^{\mathbb{R}}, \{u, v\} \in E(G)\}$ and we set,

$$\begin{aligned}&\mathcal{T}(G) \coloneqq G, \quad \text{and}\\ &\mathcal{V}(G, \{u, v\}) \coloneqq \{u, v\} \cup \big(N_G^k(u) \cap N_G^k(v)\big).\end{aligned}$$

The pooling function is then given by

$$\Psi(F(G, \{u, v\})) \coloneqq \Big(\boldsymbol{h}_{\mathcal{T}(G)}^{(L)}(u) \odot \boldsymbol{h}_{\mathcal{T}(G)}^{(L)}(v), \sum_{x\in N_G^k(u)\cap N_G^k(v)} \boldsymbol{h}_{\mathcal{T}(G)}^{(L)}(x)\Big). \tag{4}$$

From this, we arrive at the MPNN model defined by Wang et al. [2024b, Equation (9)].

F.2 The sub-sum pooling property

Here we define a useful property of the pooling function Ψ, which generalizes sum-pooling while preserving key properties relevant to our later analysis.

Definition 15 (sub-sum pooling)**.** Let $(\mathcal{T}, \mathcal{V}, \Psi)$-MPNN$(L)$ be a generalized MPNN as previously described. We say that Ψ has the *sub-sum property* with parameter C if there exists a constant $C > 0$ such that for all $(G_1, S_1), (G_2, S_2) \in \mathcal{G}_d^{\mathbb{R}} \otimes \mathfrak{R}$, with $|\mathcal{V}(G_1, S_1)| > |\mathcal{V}(G_2, S_2)|$, and for all surjective mappings $\sigma \colon \mathcal{V}(G_1, S_1) \to \mathcal{V}(G_2, S_2) \cup \{*\}$ satisfying

$$u_1 \neq u_2, \quad \sigma(u_1) = \sigma(u_2) \implies \sigma(u_1) = \sigma(u_2) = *, \tag{5}$$

we have

$$\|\Psi(F(G_1, S_1)) - \Psi(F(G_2, S_2))\|_2 \leq C \cdot \sum_{x\in\mathcal{V}(G_1,S_1)} \|\boldsymbol{h}_{\mathcal{T}(G_1)}^{(L)}(x) - \boldsymbol{h}_{\mathcal{T}(G_2)}^{(L)}(\sigma(x))\|_2, \tag{6}$$

where $\boldsymbol{h}_{\mathcal{T}(G_2)}^{(L)}(*) \coloneqq 0_{\mathbb{R}^d}$. In the case where $|\mathcal{V}(G_1, S_1)| = |\mathcal{V}(G_2, S_2)|$, we replace σ with a bijection $\mathcal{V}(G_1, S_1) \to \mathcal{V}(G_2, S_2)$.

Additionally, for any two sets A, B with $|A| > |B|$, we refer to a function $A \to B \cup \{*\}$ satisfying Equation (5) as an *extended bijection* between A and B.

The sub-sum property plays a central role in our theoretical analysis. The above architectures employ pooling functions such as sum aggregation, Hadamard product, and concatenation. We now show that each of these pooling operators satisfies the sub-sum property.

Proposition 16. Let Ψ_1, Ψ_2, and Ψ_3 be pooling functions defined as follows,

$$\begin{aligned}\Psi_1(F(G,\mathcal{V}(G))) &\coloneqq \sum_{x\in\mathcal{V}(G)} \boldsymbol{h}_G^{(L)}(x),\\ \Psi_2(F(G,\{x,y\})) &\coloneqq \boldsymbol{h}_G^{(L)}(x)\odot\boldsymbol{h}_G^{(L)}(y),\\ \Psi_3(F(G,\{x,y\})) &\coloneqq (\boldsymbol{h}_G^{(L)}(x),\boldsymbol{h}_G^{(L)}(y)).\end{aligned}$$

Then Ψ_1, Ψ_2, and Ψ_3 satisfy the sub-sum property for all $(G_1,S_1),(G_2,S_2)\in\mathcal{G}_d^{\mathbb{R}}\otimes\mathfrak{R}$, for a suitable choice of $\mathfrak{R}$.

Proof. We separate the proof into two steps: First, we show that sum aggregation and the Hadamard product have the sub-sum property. Then, we conclude the same result for the concatenation operator.

Let Ψ_1 be the sum aggregation pooling. Further, we consider σ an extended bijection as shown in Definition 15. Then the sum aggregation can be applied to Equation 6:

$$\|\Psi_1(F(G_1,S_1))-\Psi_1(F(G_2,S_2))\|_2 = \|\sum_{x\in\mathcal{V}(G_1,S_1)}\boldsymbol{h}_{G_1}^{(L)}(x)-\sum_{x'\in\mathcal{V}(G_2,S_2)}\boldsymbol{h}_{G_2}^{(L)}(x')\|_2$$

Since we know σ to be an extended bijection or a bijection depending on $\mathcal{V}(G_1,S_1),\mathcal{V}(G_2,S_2)$ it follows:

$$\begin{aligned}\|\sum_{x\in\mathcal{V}(G_1,S_1)}\boldsymbol{h}_{G_1}^{(L)}(x)-\sum_{x'\in\mathcal{V}(G_2,S_2)}\boldsymbol{h}_{G_2}^{(L)}(x')\|_2 &= \|\sum_{x\in\mathcal{V}(G_1,S_1)}\boldsymbol{h}_{G_1}^{(L)}(x)-\boldsymbol{h}_{G_2}^{(L)}(\sigma(x))\|_2\\ &\leq \sum_{x\in\mathcal{V}(G_1,S_1)}\|\boldsymbol{h}_{G_1}^{(L)}(x)-\boldsymbol{h}_{G_2}^{(L)}(\sigma(x))\|_2\end{aligned}$$

The last inequality results from applying the triangle inequality to the sum. From this, the sub-sum property directly follows with $C=1$.

For the Hadamard product $\odot$ we consider additional notation first. Given an encoding vector $\boldsymbol{h}_{G_1}^{(L)}(x)\in\mathbb{R}^n$ we denote the i-th element of such vector with x_i. We further note the following inequality:

$$|x_iy_i-x_i'y_i'| = |(x_i-x_i')y_i+x_i'(y_i-y_i')|\leq|x_i-x_i'||y_i|+|y_i-y_i'||x_i'|$$

Using this inequality, we can derive an expression for $|x_iy_i-x_i'y_i'|^2$:

$$\begin{aligned}|x_iy_i-x_i'y_i'|^2 &\leq |x_i-x_i'|^2|y_i|^2+|y_i-y_i'|^2|x_i'|^2+2|x_i||y_i||x_i-x_i'||y_i-y_i'|\\ &\leq 2|y_i|^2|x_i-x_i'|^2+2|x_i'|^2|y_i-y_i'|^2\end{aligned}$$

We now consider the Hadarmard product for $\mathcal{V}(G_1,S_1)=\{x,y\}$, $\mathcal{V}(G_2,S_2)=\{x',y'\}$ and σ as an extended bijection mapping x to x' and y to y'. Further $\mathcal{T}$ denotes a transformation function:

$$\|\Psi_2(F(G_1,S_1))-\Psi_2(F(G_2,S_2))\|_2 = \|\boldsymbol{h}_{\mathcal{T}(G_1)}^{(L)}(x)\odot\boldsymbol{h}_{\mathcal{T}(G_1)}^{(L)}(y)-\boldsymbol{h}_{\mathcal{T}(G_2)}^{(L)}(x')\odot\boldsymbol{h}_{\mathcal{T}(G_2)}^{(L)}(y')\|_2$$

$$\|\boldsymbol{h}_{\mathcal{T}(G_1)}^{(L)}(x)\odot\boldsymbol{h}_{\mathcal{T}(G_1)}^{(L)}(y)-\boldsymbol{h}_{\mathcal{T}(G_2)}^{(L)}(x')\odot\boldsymbol{h}_{\mathcal{T}(G_2)}^{(L)}(y')\|_2^2 = \sum_{i=1}^{n}(x_iy_i)^2-\sum_{i=1}^{n}(x_i'y_i')^2=\sum_{i=1}^{n}|x_iy_i-x_i'y_i'|^2$$

With our previously obtained result for $|x_iy_i-x_i'y_i'|^2$ we derive an upper bound for the Hadarmard product difference, assuming $\|\boldsymbol{h}_G^{(L)}(x)\|_\infty\leq b$ for some $b>0$:

$$\|\boldsymbol{h}_{\mathcal{T}(G_1)}^{(L)}(x)\odot\boldsymbol{h}_{\mathcal{T}(G_1)}^{(L)}(y)-\boldsymbol{h}_{\mathcal{T}(G_2)}^{(L)}(x')\odot\boldsymbol{h}_{\mathcal{T}(G_2)}^{(L)}(y')\|_2^2\leq\sum_{i=1}^{n}2|y_i|^2|x_i-x_i'|^2+2|x_i'|^2|y_i-y_i'|^2$$

$$=2\sum_{i=1}^{n}|y_i|^2|x_i-x_i'|^2+|x_i'|^2|y_i-y_i'|^2\leq 2b(\|\boldsymbol{h}_{\mathcal{T}(G_1)}^{(L)}(x)-\boldsymbol{h}_{\mathcal{T}(G_2)}^{(L)}(x')\|_2^2+\|\boldsymbol{h}_{\mathcal{T}(G_1)}^{(L)}(y)-\boldsymbol{h}_{\mathcal{T}(G_1)}^{(L)}(y')\|_2^2)$$

This implies the following sub-sum property for the Hadarmard product with $C = \sqrt{2b}$:

$$\|\boldsymbol{h}^{(L)}_{\mathcal{T}(G_1)}(x) \odot \boldsymbol{h}^{(L)}_{\mathcal{T}(G_1)}(y) - \boldsymbol{h}^{(L)}_{\mathcal{T}(G_2)}(x') \odot \boldsymbol{h}^{(L)}_{\mathcal{T}(G_2)}(y')\|_2$$
$$\leq \sqrt{2b}(\|\boldsymbol{h}^{(L)}_{\mathcal{T}(G_1)}(x) - \boldsymbol{h}^{(L)}_{\mathcal{T}(G_2)}(x')\|_2 + \|\boldsymbol{h}^{(L)}_{\mathcal{T}(G_1)}(y) - \boldsymbol{h}^{(L)}_{\mathcal{T}(G_1)}(y')\|_2)$$

In the last step, we consider the concatenation operator Ψ_3. Since the concatenation of two vectors $x, y \in \mathbb{R}^n$ is given by $(x_1, \dots, x_n, y_1, \dots, y_n) = (x, y) \in \mathbb{R}^{2n}$ we can separate the Euclidean norm:

$$\|(x, y) - (x', y')\|_2 = \|(x - x') + (y - y')\|_2 \leq \|x - x'\|_2 + \|y - y'\|_2$$

From this, the sub-sum property directly follows. □

Since the Hadamard product and the concatenation operator have the sub-sum property, it is easy to verify that the property also holds for the pooling function used in neural common neighbors Equation (4).

G The 1-dimensional Weisfeiler–Leman algorithm

The 1-*dimensional Weisfeiler–Leman algorithm* (1-WL) or *color refinement* is a well-studied heuristic for the graph isomorphism problem, originally proposed by Weisfeiler and Leman [1968].[2] Intuitively, the algorithm determines if two graphs are non-isomorphic by iteratively coloring or labeling nodes. Given an initial coloring or labeling of the nodes of both graphs, e.g., their degree or application-specific information, in each iteration, two nodes with the same label get different labels if the number of identically labeled neighbors is unequal. These labels induce a node 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 nodes annotated with a specific label differs 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, a_G) be a node-featured graph. In each iteration, $t > 0$, the 1-WL computes a *node coloring* $C^1_t \colon V(G) \to \mathbb{N}$, depending on the coloring of the neighbors. That is, in iteration $t > 0$, we set

$$C^1_t(v) := \mathsf{RELABEL}\Big(\big(C^1_{t-1}(v), \{\!\{C^1_{t-1}(u) \mid u \in N(v)\}\!\}\big)\Big),$$

for node $v \in V(G)$, 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 $C^1_0 := a_G$ is used.[3] 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 nodes 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^1_t and C^1_{i+t} are equal, or, equivalently,

$$C^1_t(v) = C^1_t(w) \iff C^1_{t+1}(v) = C^1_{t+1}(w),$$

for all nodes $v, w \in V(G \dot\cup H)$, then the algorithm terminates. For such t, we define the *stable coloring* $C^1_\infty(v) = C^1_t(v)$, for $v \in V(G \dot\cup H)$. The stable coloring is reached after at most $\max\{|V(G)|, |V(H)|\}$ iterations [Grohe, 2017].

Unrollings Following Morris et al. [2020b], given a (node)-featured graph (G, a_G), we define the *unrolling tree* of depth $L \in \mathbb{N}_0$ for a node $u \in V(G)$, denoted as $\mathsf{unr}(G, u, L)$, inductively as follows.

1. For $L = 0$, we consider the trivial tree as an isolated node with feature $a_G(u)$.
2. For $L > 0$, we consider the root node with label $a_G(u)$ and, for $v \in N(u)$, we attach the subtree $\mathsf{unr}(G, v, L - 1)$ under the root.

The above unrolling tree construction characterizes the 1-WL algorithm through the following lemma.

[2] 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 nodes in a given graph; see Grohe [2021] for details. Following customs in the machine learning literature, we consider both algorithms equivalent.

[3] Here, we implicitly assume an injective function from Σ to $\mathbb{N}$.

Lemma 17 (Folklore, see, e.g., Morris et al. [2020a]). The following are equivalent for $L \in \mathbb{N}_0$, given a featured graph (G, a_G) and nodes $u, v \in V(G)$.

1. The nodes u and v have the same color after L iterations of the 1-WL.
2. The unrolling trees $\mathsf{unr}(G, u, L)$ and $\mathsf{unr}(G, v, L)$ are isomorphic. $\square$

Note that for an edge-featured graph, we can analogously define a 1-WL-variant and an unrolling tree (with edge features), satisfying a similar version of Lemma 17.

H Learning frameworks

This appendix fully formalizes the inductive and transductive learning frameworks introduced in Section 2.2. We specify the underlying probability spaces, data generation models, symmetry assumptions, and loss functions that underpin our generalization analysis. The distinction between these frameworks lies in the way they leverage available information. Inductive learning involves training models to infer general patterns from observed data, enabling predictions on unseen instances. This ability to generalize beyond the training set is at the core of most supervised learning methods. In contrast, transductive learning focuses on deriving predictions directly for specific, unlabeled data points, as seen in semi-supervised and few-shot learning, without constructing an explicit general model. In the subsequent sections, we formalize the statistical learning framework for any machine learning task, define the generalization error in both settings, and describe how node and link prediction tasks can be addressed using either an inductive or transductive approach.

Inductive statistical learning Let $(\Omega, \mathcal{F}, P)$ be a probability space $\mathcal{X}$ be an arbitrary input space and $\mathcal{Y}$ the label space. For example, in *binary classification*, the label set is typically represented as $\{0, 1\}$. Let $\mathcal{Z}$ denote the product space $\mathcal{X} \times \mathcal{Y}$. In supervised learning, we have access to a finite set of *training points*,

$$\mathcal{S} \coloneqq \{(x, y) \mid (x, y) \in \mathcal{X} \times \mathcal{Y}\},$$

where each $(x_i, y_i)_{i=1}^N$ is sampled according to the same distribution μ on $\mathcal{Z}$, and $|\mathcal{S}| = N \in \mathbb{N}$. In the special case where each $(x_i, y_i) \in \mathcal{S}$ is independently drawn, we say that we have an *i.i.d. sampling process*.

We consider a family of mappings from $\mathcal{X}$ to $\mathcal{Y}$, denoted by $\mathcal{H}$, often referred to as the *hypothesis class*, and its elements as *concepts*. To formally measure the quality of a concept, we define a *loss function* $\ell\colon \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ bounded by some $M \in \mathbb{N}$. The goal then is to choose a function $h \in \mathcal{H}$ that minimizes the expectation $\mathbb{E}_\mu[\ell(h, X, Y)]$.

A *learning algorithm* $\mathcal{A}$ is a procedure that takes the training data $\mathcal{S}$ as input and outputs a concept $h \in \mathcal{H}$; we usually denote $h_\mathcal{S} \coloneqq \mathcal{A}_\mathcal{S}$. Hence, the chosen concept $h \in \mathcal{H}$ also depends on the training data $\mathcal{S}$. For a given sample $\mathcal{S}$ of cardinality N, we write

$$\ell_{\text{exp}}(\mathcal{A}_\mathcal{S}) \coloneqq \mathbb{E}_\mu(\ell(\mathcal{A}_\mathcal{S}, X, Y)),$$

to denote the *expected risk*. However, since the underlying distribution μ is unknown, we usually resort to minimizing the *empirical risk* for a given sample $\mathcal{S}$, i.e.,

$$\ell_{\text{emp}}(\mathcal{A}_\mathcal{S}) \coloneqq \frac{1}{N} \sum_{(x,y) \in \mathcal{S}} \ell(\mathcal{A}_\mathcal{S}, x, y).$$

Given the above notation, the *generalization error* of a hypothesis $h \in \mathcal{H}$ is defined as the absolute difference of the above two quantities, i.e.,

$$|\ell_{\text{emp}}(h) - \ell_{\text{exp}}(h)|.$$

Transductive statistical learning Again, let $(\Omega, \mathcal{F}, P)$ be a probability space. As in the inductive setting, we are given pairs (x, y) of labeled data where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, for some arbitrary input space $\mathcal{X}$ and label space $\mathcal{Y}$ and we similarly define a hypothesis class $\mathcal{H}$ and a loss function ℓ. However, in the transductive setting, we do not assume any unknown underlying distribution for generating the data, as we know from the beginning the inputs for which we want to predict labels, allowing us to utilize them throughout the learning process.

Thus, we consider the space $\mathcal{Z} \coloneqq \{(x_i, y_i)\}_{i=1}^{m+u}$, where $n = m + u$, consisting of pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$. We can access m labeled pairs and the remaining u unlabeled inputs. The objective is to choose a function $h \in \mathcal{H}$ that minimizes the average loss on the u unlabeled samples.

Since we have not assumed an underlying distribution, we must somehow introduce randomness into our sampling process. Following Vapnik [2006, Setting 1], we model the sampling process by assuming that the m labeled training examples are sampled *uniformly without replacement* from the whole space $\mathcal{Z}$.

We may assume that we first choose a permutation $\pi \colon [n] \to [n]$ uniformly at random from the set of all possible $n!$ permutations. We then order the space $\mathcal{Z}$ according to this permutation, obtaining

$$\mathcal{Z}_\pi = (x_{\pi(i)}, y_{\pi(i)})_{i=1}^{m+u}.$$

We select the first m elements from this ordered sequence as our labeled training set and the remaining u elements as the unlabeled test set. Thus, our training set is defined as

$$\mathcal{S}_\pi = \{(x_{\pi(i)}, y_{\pi(i)})\}_{i=1}^{m} \cup \{x_{\pi(i)}\}_{i=m+1}^{m+u}.$$

The only source of randomness in this setting comes from the random variable $\pi \colon \Omega \to S_n$, where $S_n = \{\rho \colon [n] \to [n] \mid \rho \text{ is a bijection}\}$ denotes the symmetric group, with

$$P(\pi \in A) = \frac{|A|}{n!}, \quad \text{for } A \subset S_n.$$

We write $\pi \sim \text{Unif}(S_n)$. A *transductive learning algorithm* $\mathcal{A}$ is then defined as a function $S_n \to \mathcal{H}$, and the generalization gap is given by the difference

$$|R_m - R_u|,$$

where,

$$R_m := \frac{1}{m}\sum_{i=1}^{m} \ell(\mathcal{A}_\pi, x_{\pi(i)}, y_{\pi(i)}) \quad \text{and} \quad R_u := \frac{1}{u}\sum_{i=m+1}^{m+u} \ell(\mathcal{A}_\pi, x_{\pi(i)}, y_{\pi(i)}),$$

where $\mathcal{A}_\pi$ depends on π through $\mathcal{S}_\pi$.

Remarks

1. This setting differs significantly from the i.i.d. case, as the samples are not independent. Specifically, since we sample without replacement, each training sample depends on the others. Consequently, concentration inequalities for independent random variables or graph-dependent data, such as Theorem 27, are not directly applicable.
2. In this setting, the underlying distribution and dependencies between samples are explicitly known. These dependencies should be exploited to derive generalization bounds.
3. Considering our goal, it is reasonable to focus on learning algorithms that exhibit symmetry concerning the arguments $\{(x_{\pi(i)}, y_{\pi(i)})\}_{i=1}^{m}$ for any permutation $\pi \in S_n$. This symmetry ensures that the expected values satisfy the following conditions:

$$\mathbb{E}_\pi[\ell(\mathcal{A}_\pi, x_{\pi(i)}, y_{\pi(i)})] = \mathbb{E}_\pi[\ell(\mathcal{A}_\pi, x_{\pi(1)}, y_{\pi(1)})], \text{ for } i \in [m]$$
$$\mathbb{E}_\pi[\ell(\mathcal{A}_\pi, x_{\pi(m+j)}, y_{\pi(m+j)})] = \mathbb{E}_\pi[\ell(\mathcal{A}_\pi, x_{\pi(m+1)}, y_{\pi(m+1)})], \text{ for } j \in [u].$$

   Implying,

$$\mathbb{E}_\pi[R_m] = \mathbb{E}_\pi[\ell(\mathcal{A}_\pi, x_{\pi(1)}, y_{\pi(1)})],$$
$$\mathbb{E}_\pi[R_u] = \mathbb{E}_\pi[\ell(\mathcal{A}_\pi, x_{\pi(m+1)}, y_{\pi(m+1)})].$$

Learning on Graphs We specify below how the statistical learning approaches discussed earlier (inductive and transductive) apply to graph-learning tasks. More precisely, we focus on node and link prediction, the primary tasks analyzed in this work. However, the setting can be naturally extended to any graph-representation learning task.

In the inductive setting for node prediction, the input space is given by $\mathcal{X} := \mathcal{G}' \otimes V$, where $\mathcal{G}' \subset \mathcal{G}_d^{\mathbb{R}}$, and the label space is $\mathcal{Y} = \{0, 1\}$ for classification or $\mathbb{R}$ for regression. Thus, the dataset consists of triplets (G, u, y), where $G \in \mathcal{G}'$, $u \in V(G)$, and $y \in \mathcal{Y}$, sampled according to a distribution μ, as described earlier. Possible dependencies between samples will be specified based on our assumptions, as detailed in Section 4. A natural assumption regarding the dependency structure in the dataset is that two sampled triplets (G_1, u_1, y_1) and (G_2, u_2, y_2) are possibly dependent when $G_1 = G_2$. This follows from the assumption that the underlying distribution first generates the entire graph with all node labels, and then only a subset of these labels is revealed to the learner. Similarly, the input space is given by $\mathcal{X} \coloneqq \mathcal{G}' \otimes E$ for link prediction.

In the transductive setting for node prediction, we assume that we are given a fixed graph $G \in \mathcal{G}_d^{\mathbb{R}}$ with node set $V(G) = \{u_1, \ldots, u_n\}$, where $n = m+u$, and that the labels for m nodes in $V(G)$ are provided. Thus, we define the dataset as $\mathcal{Z} \coloneqq \{(G, u_i, y_i)\}_{i=1}^{m+u}$. When we say the graph is given, we implicitly assume that the initial node features are also provided. Similarly, in the transductive framework for link prediction, we define $\mathcal{Z} = \{(G, e_i, y_i)\}_{i=1}^{m+u}$, where each (G, e_i) belongs to $G \otimes E \coloneqq \{(G, e) \mid e = \{u, v\} \subset V(G)\}$

Cross-entropy loss function Finally, we note that the quantities $\ell_{\text{exp}}, \ell_{\text{emp}}, R_u$, and R_m can be similarly defined when the hypothesis class $\mathcal{H}$ consists of functions mapping from $\mathcal{X}$ to $\mathbb{R}$ (rather than $\{0, 1\}$), as long as the loss function $\ell \colon \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$ is properly defined. In this work, all L-layer generalized MPNNs are assumed to map inputs to vectors $\boldsymbol{h} \in \mathbb{R}^{d_L}$, which are then processed by a trainable feedforward neural network (FNN) producing an output $x \in \mathbb{R}$. We use the binary cross-entropy loss with a sigmoid activation to compute the loss between x and the true label $y \in \{0, 1\}$:

$$\text{BCE}(x, y) \coloneqq y \log(\sigma(x)) + (1 - y) \log(1 - \sigma(x)),$$

where $\sigma(x)$ denotes the sigmoid function. Importantly, note that when using the above setting for the loss function $\ell \colon \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$, it is easy to verify that $\ell(h, \cdot)$, satisfies the Lipschitz property as required by Theorem 3, and Theorem 4.

I The unrolling distances

Here, we formally define the unrolling distances introduced in Section 3 and prove their main properties.

Recalling the definition of unrolling trees from Appendix G, given $(G, S) \in \mathcal{G}' \otimes \mathfrak{R}$, a transformation function $\mathcal{T} \colon \mathcal{G}' \to \mathcal{G}_d^{\mathbb{R}}$, selection function $\mathcal{V} \colon (G, S) \mapsto A \subset V(\mathcal{T}(G))$, and a number of layers $L \in \mathbb{N}$, we define the following multi-set of unrolling trees,

$$\mathcal{F}^{(L)}(G, S) = \{\!\{ \mathsf{unr}(\mathcal{T}(G), u, L) \mid u \in \mathcal{V}(G, S) \}\!\}.$$

Our metric is intended to measure the alignment between the multisets of unrolling trees $\mathcal{F}^{(L)}(G_1, S_1)$ and $\mathcal{F}^{(L)}(G_2, S_2)$. We first define a padding process for handling multisets and unrolling of different sizes.

Given a rooted tree (T, r) with node features in $\mathbb{R}^d \setminus \{\mathbf{o}_{\mathbb{R}^d}\}$, and parameters $L, q \in \mathbb{N}$ such that

$$L \geq \max\{\text{length}(p) \mid v \in V(T), p \in \mathcal{P}(r, v)\}, \quad q \geq \max\{|N_{T,\text{out}}(v)| \mid v \in V(T)\},$$

where L is at least the depth of the tree, and q is at least the maximum out-degree of the corresponding directed tree, noting that any rooted tree has an implicit direction pointing away from the root, we define the (q, L)-padded rooted tree (T', r) according to the following algorithm.

1. Initialize $l = 0$ and set $T_0 = T$.
2. For each node u at level l of T_0, add children with node feature $\mathbf{o}_{\mathbb{R}^d}$ until u has exactly q children. The resulting tree is denoted as T_1.
3. If $l = L - 1$, the algorithm terminates, and we set $T' = T_1$. Otherwise, repeat step 2 with $l = l + 1$ and set $T_0 = T_1$.

We denote the resulting rooted tree T' as $\mathsf{padd}_{q,L}(T)$, where the root remains unchanged.

Now, given $(G_1, S_1), (G_2, S_2) \in (\mathcal{G}_d^{\mathbb{R}}, \mathfrak{R})$, we apply the $\mathsf{padd}_{q,L}$ process to each unrolling tree in both $\mathcal{F}^{(L)}(G_1, S_1)$ and $\mathcal{F}^{(L)}(G_2, S_2)$, with q set to the maximum node degree, considering the trees as directed trees, and L set to the maximum depth among all trees in $\mathcal{F}^{(L)}(G_1, S_1)$ and $\mathcal{F}^{(L)}(G_2, S_2)$. Ignoring node features, the resulting trees are all isomorphic, i.e., all resulting trees are L-depth-rooted trees where each non-leaf node has exactly q children. Finally, suppose the resulting multisets have different sizes. In that case, we augment the smaller one with additional trees of the same structure and node features $\mathbf{o}_{\mathbb{R}^d}$ until both multisets have the same cardinality. We denote the resulting multisets as (M_1, M_2) and define the mapping,

$$\rho \colon (\mathcal{F}^{(L)}(G_1, S_1), \mathcal{F}^{(L)}(G_2, S_2)) \mapsto (M_1, M_2).$$

See Figure 1 for an illustration of the padding process through the function ρ.

Finally, let G_A, G_B be two graphs such that an edge-preserving bijection exists between their nodes. We define

$$S(G_A, G_B) \coloneqq \{\varphi\colon V(G_A) \to V(G_B) \mid \varphi \text{ is an edge-preserving bijection}\}.$$

We now recall the definition of the unrolling distance function on

$$\mathcal{G}_d^{\mathbb{R}} \otimes \mathfrak{C} \coloneqq \{(G, S) \mid G \in \mathcal{G}_d^{\mathbb{R}}, S \subseteq V(G)\}.$$

Definition 18 (Restated, Definition 14)**.** Given a transformation function $\mathcal{T}\colon \mathcal{G}' \to \mathcal{G}_d^{\mathbb{R}}$, and a selection function $\mathcal{V}\colon (G, S) \mapsto A \subset V(\mathcal{T}(G))$, for $(G, S) \in \mathcal{G}' \otimes \mathfrak{R}$, we define the $(\mathcal{T}, \mathcal{V})$-*Unrolling Distance* ($\mathrm{UD}_{\mathcal{T},\mathcal{V}}^{(L)}$) of depth L on $\mathcal{G}_d^{\mathbb{R}} \otimes \mathfrak{C}$ as follows. Let $(G_1, S_1), (G_2, S_2) \in \mathcal{G}' \otimes \mathfrak{C}$ and $\rho((\mathcal{F}^{(L)}(G_1, S_1), \mathcal{F}^{(L)}(G_2, S_2))) = (M_1, M_2)$. Then,

$$\mathrm{UD}_{\mathcal{T},\mathcal{V}}^{(L)}((G_1, S_1), (G_2, S_2)) = \min_{\varphi \in S(F_1, F_2)} \sum_{x \in V(F_1)} \|a_{F_1}(x) - a_{F_2}(\varphi(x))\|_2,$$

where F_i is the forest consisting of the disjoint union of all trees in M_i, for $i = 1, 2$, and we recall that

$$S(G, H) \coloneqq \{\varphi\colon V(G) \to V(H) \mid \varphi \text{ is an edge-preserving bijection}\}.$$

Let T_1, T_2 be two rooted trees such that both have depth L and all non-leaf nodes in T_1 and T_2 have exactly q children. Let $\overrightarrow{T_1}$ and $\overrightarrow{T_2}$ be the directed versions of these trees. Comparing the degrees of the nodes (for T_1, T_2), it is easy to verify that $S(T_1, T_2) = S(\overrightarrow{T_1}, \overrightarrow{T_2})$, implying that in the above definition, it does not matter whether we consider the padded unrollings as directed or undirected trees.

We now prove that the Unrolling Distance is a well-defined pseudo-metric on $\mathcal{G}_d^{\mathbb{R}} \otimes \mathfrak{C}$.

Lemma 19. For every $L \in \mathbb{N}$, the Unrolling Distance $\mathrm{UD}_{\mathcal{T},\mathcal{V}}^{(L)}$ in Equation (2) is a well-defined pseudo-metric on $\mathcal{G}_d^{\mathbb{R}} \otimes \mathfrak{C}$.

Proof. The proof follows the reasoning of the proof in Bento and Ioannidis [2019, Lemma 1]. Let $\mathcal{G}'$ denote the set of all graphs in $\mathcal{G}_d^{\mathbb{R}}$ that do not contain nodes with zero feature vectors. That is, $\mathcal{G}' = \mathcal{G}_d^{\mathbb{R}} \cap \{G \in \mathcal{G}_d^{\mathbb{R}} \mid a_G(u) \neq \mathbf{o}, \forall u \in V(G)\}$. We prove the pseudo-metric property on $\mathcal{G}' \otimes \mathfrak{C}$ for simplicity of notation, but it can easily be extended to $\mathcal{G}_d^{\mathbb{R}} \otimes \mathfrak{C}$.

Let $(G_i, S_i) \in \mathcal{G}' \otimes \mathfrak{R}$. We denote by F_i the forest consisting of all padded unrolling trees of depth L of graph G_i as defined in the definition of the Unrolling Distance and $a_{F_i}(u)$, being the node feature of node u in $V(F_i)$. Note that we can assume that F_1, F_2, F_3 have the same structure, since if that is not the case, we can pad them by adding nodes (or even trees) with $\mathbf{o}_{\mathbb{R}^d}$ features. We further assume for simplicity of notation that $V(F_i) = [n]$, for $i \in [3]$.

Now, we define the distance matrices $\boldsymbol{L}^{(\kappa,\lambda)}$, as $\boldsymbol{L}_{i,j}^{(\kappa,\lambda)} = \|a_{F_\kappa}(i) - a_{F_\lambda}(j)\|_2$, for $\kappa, \lambda, i, j \in [3]$. Let also $\mathcal{P}_{i,j}$, being the set of all permutation $n \times n$ matrices $\boldsymbol{P}$ satisfying $\boldsymbol{A}(F_i)\boldsymbol{P} = \boldsymbol{P}\boldsymbol{A}(F_j)$, for $i, j \in [3]$. Hence, the unrolling distance can be rewritten as,

$$\mathrm{UD}_{\mathcal{T},\mathcal{V}}^{(L)}((G_1, S_2), (G_2, S_2)) = \min_{\boldsymbol{P} \in \mathcal{P}_{1,2}} \Big(\mathrm{Tr}(\boldsymbol{P}^{\mathrm{T}} \boldsymbol{L}^{(1,2)})\Big). \tag{7}$$

It is easy to observe from Equation (7) that the identity and symmetry properties are trivially satisfied. For the triangle inequality, let $\boldsymbol{P}^{(i,j)} \in \mathcal{P}_{i,j}$ be the permutation matrix that minimizes $\mathrm{UD}_{\mathcal{T},\mathcal{V}}^{(L)}((G_i, S_i), (G_j, S_j))$. Note that $\boldsymbol{P}^{(1,2)}\boldsymbol{P}^{(2,3)} \in \mathcal{P}_{1,3}$, that is because permutation matrices are closed under products and

$$\boldsymbol{A}(F_1)\boldsymbol{P}^{(1,2)}\boldsymbol{P}^{(2,3)} = \boldsymbol{P}^{(1,2)}\boldsymbol{A}(F_2)\boldsymbol{P}^{(2,3)} = \boldsymbol{P}^{(1,2)}\boldsymbol{P}^{(2,3)}\boldsymbol{A}(F_3).$$

Therefore,

$$\mathrm{UD}_{\mathcal{T},\mathcal{V}}^{(L)}((G_1, S_1), (G_3, S_3)) \leq \mathrm{Tr}((\boldsymbol{P}^{(1,2)}\boldsymbol{P}^{(2,3)})^T \boldsymbol{L}^{(1,3)}).$$

It is, therefore, sufficient to show that,

$$\mathrm{Tr}\Big((\boldsymbol{P}^{(1,2)}\boldsymbol{P}^{(2,3)})^{\mathrm{T}} \boldsymbol{L}^{(1,3)}\Big) \leq \mathrm{Tr}\Big(\Big(\boldsymbol{P}^{(1,2)}\Big)^T \boldsymbol{L}^{(1,2)}\Big) + \mathrm{Tr}\Big(\Big(\boldsymbol{P}^{(2,3)}\Big)^T \boldsymbol{L}^{(2,3)}\Big).$$

To verify this, we use the following property of the trace operator. For $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^{n \times n}$, we have that,

$$\operatorname{Tr}((\boldsymbol{A}\boldsymbol{B})^T \boldsymbol{C}) = \sum_{i,j,k=1}^{n} A_{i,k} B_{k,j} C_{i,j}.$$

Applying this we have that:

$$\begin{aligned}
\operatorname{Tr}((\boldsymbol{P}^{(1,2)}\boldsymbol{P}^{(2,3)})^T \boldsymbol{L}^{(1,3)}) &= \sum_{i,j,k=1}^{n} P_{i,k}^{(1,2)} P_{k,j}^{(2,3)} \|a_{F_1}(i) - a_{F_3}(j)\|_2 \\
&\leq \sum_{i,j,k=1}^{n} P_{i,k}^{(1,2)} P_{k,j}^{(2,3)} (\|a_{F_1}(i) - a_{F_2}(k)\|_2 + \|a_{F_2}(k) - a_{F_3}(j)\|_2) \\
&= \sum_{i,k=1}^{n} P_{i,k}^{(1,2)} \|a_{F_1}(i) - a_{F_2}(k)\|_2 \underbrace{\sum_{j=1}^{n} P_{k,j}^{(2,3)}}_{=1} \\
&+ \sum_{k,j=1}^{n} P_{k,j}^{(2,3)} \|a_{F_2}(k) - a_{F_3}(j)\|_2 \underbrace{\sum_{i=1}^{n} P_{i,k}^{(1,2)}}_{=1} \\
&= \operatorname{Tr}\left(\left(\boldsymbol{P}^{(1,2)}\right)^T \boldsymbol{L}^{(1,2)}\right) + \operatorname{Tr}\left(\left(\boldsymbol{P}^{(2,3)}\right)^T \boldsymbol{L}^{(2,3)}\right)
\end{aligned}$$

□

Finally, note that the above definition can be generalized by introducing different weights on each level of the unrolling trees. Given a forest of rooted trees F, define the level function $l_F \colon V(F) \to \mathbb{N}$, where $l_F(u)$ is the level of the tree node u belongs to. Given a weight function $\omega \colon \mathbb{N} \to \mathbb{R}^+$, we define the following *Weighted Unrolling Distance*,

$$\operatorname{UD}_{\mathcal{T},\mathcal{V},\omega}^{(L)}((G_1, S_2), (G_2, S_2)) = \min_{\varphi \in S(F_1,F_2)} \sum_{x \in V(F_1)} \omega(l_{F_1}(x)) \|a_{F_1}(x) - a_{F_2}(\varphi(x))\|_2. \quad (8)$$

To verify that Equation (8) is a well-defined pseudo-metric, we would have to modify the definition of unrolling trees by weighting the node features based on their level on the tree and then follow the proof of Lemma 19. Regarding the numerical computation of the above distance, we follow the optimal transport approach described in Chuang and Jegelka [2022], which formulates the problem as an instance of optimal transport and solves it using dynamic programming. Alternatively, faster approximation methods, such as those proposed by Altschuler et al. [2017], can be employed.

The following result shows that the sub-sum property, see Definition 15, provides a sufficient condition for a generalized $(\mathcal{T}, \mathcal{V}, \Psi)$-MPNN$(L)$ to satisfy the Lipschitz property with respect to the $\operatorname{UD}_{\mathcal{T},\mathcal{V},\omega}^{(L)}$ pseudo-metric, for a specific weighting function ω.

Proposition 20. Let $\mathcal{G}_d^{\mathbb{R}} \otimes \mathfrak{R}$ be a representation task, and let $(\mathcal{T}, \mathcal{V}, \Psi)$ be a generalized MPNN(L) with an MPNN architecture given by Equation (1), where Ψ satisfies the sub-sum property with parameter $\widetilde{C}$. Then, for

$$\omega \colon l \mapsto \binom{L}{l}, \quad l \in \{0, \ldots, L\},$$

where $\binom{L}{l}$ denotes the binomial coefficient. The generalized MPNN is Lipschitz concerning the weighted unrolling distance $\operatorname{UD}_{\mathcal{T},\mathcal{V},\omega}^{(L)}$. That is, there exists a constant $C > 0$ such that for all $(G_1, S_1), (G_2, S_2) \in \mathcal{G}_d^{\mathbb{R}} \otimes \mathfrak{R}$, we have

$$\|h_{\mathcal{T},\mathcal{V},\Psi}(G_1, S_1) - h_{\mathcal{T},\mathcal{V},\Psi}(G_2, S_2)\|_2 \leq C \cdot \operatorname{UD}_{\mathcal{T},\mathcal{V},\omega}^{(L)}((G_1, S_1), (G_2, S_2)).$$

Moreover, the constant C depends on L, $\widetilde{C}$, the Lipschitz constants L_{φ_t} and the bound B on the 2-norm of weighted matrices described in Equation (1).

Proof. Let $(G_1, S_1), (G_2, S_2) \in \mathcal{G}_d^{\mathbb{R}}$ (without $\mathbf{o}$ node features), and let F_1, F_2 be their corresponding padded forests as described in Definition 14. First, note that each edge-preserving bijection φ between F_1 and F_2 (i.e., $\varphi \in S(F_1, F_2)$) induces a sequence of bijections $\bar{\sigma}_l \colon V_1^{(l)} \to V_2^{(l)}$, where $V_1^{(l)}$ and $V_2^{(l)}$ denote the sets of nodes in F_1 and F_2 at depth l, respectively, for $l \in \{0, \ldots, L\}$.

Based on $\bar{\sigma}_l$, we can define a sequence of extended bijections σ_l between $V_1^{(l)} \cap \{u \in V_1^{(l)} \mid a_{F_1}(u) \neq \mathbf{o}\}$ and $V_2^{(l)} \cap \{u \in V_2^{(l)} \mid a_{F_2}(u) \neq \mathbf{o}\}$ as follows:

$$\sigma_l(u) = \begin{cases} \bar{\sigma}_l(u), & \text{if } a_{F_1}(u) \neq \mathbf{o}, \\ *, & \text{otherwise.} \end{cases}$$

A similar definition applies when $|V_1^{(l)}| < |V_2^{(l)}|$. Therefore, each $\varphi \in S(F_1, F_2)$ uniquely induces a sequence of extended bijections. It is straightforward to verify that this sequence is unique and that this process can be reversed, meaning that each sequence of extended bijections induces a unique $\varphi \in S(F_1, F_2)$. For simplicity, we assume $B = 1$ (the proof follows similarly for general B). Finally, by the construction of the unrolling trees, each node in $V_i^{(l)} \cap \{u \in V_1^{(l)} \mid a_{F_1}(u) \neq \mathbf{o}\}$ can be mapped to the corresponding node $u \in V(\mathcal{T}(G_i))$, for $i = 1, 2, l = 0, \ldots, L$. Therefore, with a slight abuse of notation, σ_l also defines an extended bijection between nodes in $\mathcal{T}(G_1)$ and $\mathcal{T}(G_2)$ that can be found in the l-th level of some tree.

Let $\varphi \in S(F_1, F_2)$, and let $\sigma_0, \ldots, \sigma_L$ be the corresponding induced extended bijections. We define $\delta(L, l)$ as follows [4]:

$$\begin{aligned} \delta(L, 0) &= \sum_{x_0 \in \mathcal{V}(G_1, S_1)} \|h^{(L)}_{\mathcal{T}(G_1)}(x_0) - h^{(L)}_{\mathcal{T}(G_2)}(\sigma_0(x_0))\|_2, \\ \delta(L, l) &= \sum_{x_0 \in \mathcal{V}(G_1, S_1)} \sum_{x_1 \sim x_0} \cdots \sum_{x_l \sim x_{l-1}} \|h^{(L)}_{\mathcal{T}(G_1)}(x_l) - h^{(L)}_{\mathcal{T}(G_2)}(\sigma_l(x_l))\|_2, \quad \text{for } l \in [L], \end{aligned}$$

where $h^{(l)}_{\mathcal{T}(G_i)}(*) = \mathbf{o}$, for all $i = 1, 2$, $l \in \{0, \ldots, L\}$ and $x_l \sim x_{l-1} \Leftrightarrow x_l \in N_{\mathcal{T}(G_1)}(x_{l-1})$, for all $l \in \{0, \ldots, L\}$.

It is easy to observe that,

$$\begin{aligned} \delta(L, l) \leq L_{\varphi_L} & \sum_{x_0 \in \mathcal{V}(G_1, S_1)} \sum_{x_1 \sim x_0} \cdots \sum_{x_l \sim x_{l-1}} \|\boldsymbol{W}_L^{(1)}\| \|h^{(L-1)}_{\mathcal{T}(G_1)}(x_l) - h^{(L-1)}_{\mathcal{T}(G_2)}(\sigma_l(x_l))\|_2 \\ & + L_{\varphi_L} \sum_{x_0 \in \mathcal{V}(G_1, S_1)} \sum_{x_1 \sim x_0} \cdots \sum_{x_{l+1} \sim x_l} \|\boldsymbol{W}_L^{(2)}\| \|h^{(L-1)}_{\mathcal{T}(G_1)}(x_l) - h^{(L-1)}_{\mathcal{T}(G_2)}(\sigma_{l+1}(x_{l+1}))\|_2 \\ & \leq L_{\varphi_L} \delta(L-1, l) + L_{\varphi_L} \delta(L-1, l+1) \end{aligned}$$

Thus, recursively applying this bound, we get:

$$\begin{aligned} \|\varPsi(F(G_1, S_1)) - \varPsi(F(G_2, S_2))\|_2 &\leq \widetilde{C} \cdot \sum_{x \in \mathcal{V}(G_1, S_1)} \|h^{(L)}_{\mathcal{T}(G_1)}(x) - h^{(L)}_{\mathcal{T}(G_2)}(\sigma(x))\|_2 \\ &= \widetilde{C} \delta(L, 0) \\ &\leq \widetilde{C} L_{\varphi_L} (\delta(L-1, 0) + \delta(L-1, 1)) \\ &\leq \cdots \\ &\leq \widetilde{C} \left(\prod_{t=1}^{L} L_{\varphi_t} \right) \sum_{l=0}^{L-1} \binom{L}{l} \delta(0, l) \\ &= \widetilde{C} \left(\prod_{t=1}^{L} L_{\varphi_t} \right) \cdot \mathrm{UD}^{(L)}_{\mathcal{T}, \mathcal{V}, \omega}((G_1, S_1), (G_2, S_2)). \end{aligned}$$

□

[4] In this notation, without loss of generality and for simplicity, we assume that $|N_{F_1}(x)| \geq |N_{F_2}(y)|$. If this assumption does not hold, the notation can be trivially adjusted to reflect the reverse case.

J Extended robustness generalization under dependency

Xu et al. [2021] derived generalization bounds for the inductive setting under the i.i.d. assumption, based on the covering number of the product space of inputs and outputs concerning a suitable pseudo-metric. In the following, we extend the results of Xu and Mannor [2012] to settings with dependent data. We consider both an inductive case with known dependency structure and a transductive case with explicit dependencies. Below we define a modified notion of robustness following Xu and Mannor [2012], which we refer to as *uniform robustness*. In what follows $\mathcal{N}(\mathcal{X}, d, \varepsilon)$ denotes the covering number of the (pseudo)metric space $(\mathcal{X}, d)$ with radius ε.

Definition 21 (Uniform robustness)**.** We say that a learning algorithm $\mathcal{A}$ on the space $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$ with hypothesis class $\mathcal{H}$ is (K, ε)*-uniformly robust* if there exist constants $K \in \mathbb{N}$ and $\varepsilon > 0$ such that, there exists a partition $\{C_i\}_{i=1}^{K}$ of $\mathcal{Z}$ satisfying the following property: For all $i \in [K]$, and for any sample $\mathcal{S}$,

$$z_1, z_2 \in C_i \implies |\ell(\mathcal{A}_\mathcal{S}, z_1) - \ell(\mathcal{A}_\mathcal{S}, z_2)| \leq \varepsilon.$$

In the spirit of Xu and Mannor [2012, Theorem 14], the following theorem shows that Lipschitz continuity of the loss function concerning a pseudo-metric d is sufficient for a hypothesis class to satisfy the robustness property. It also provides robustness parameters regarding the covering number of the space concerning d and the associated radius. The proof follows from Xu and Mannor [2012, Theorem 14].

Theorem 22. Let $\mathcal{A}$ be a learning algorithm on $\mathcal{Z}$ for a hypothesis class $\mathcal{H}$, and let $\ell \colon \mathcal{H} \times \mathcal{Z} \to \mathbb{R}$ be a loss function. Suppose d is a pseudo-metric on $\mathcal{Z}$. If $\ell(h, \cdot)$ is C-Lipschitz with respect to d, i.e., for all samples $\mathcal{S}$ and $z_1, z_2 \in \mathcal{Z}$,

$$|\ell(\mathcal{A}_\mathcal{S}, z_1) - \ell(\mathcal{A}_\mathcal{S}, z_2)| \leq C \cdot d(z_1, z_2),$$

then $\mathcal{A}$ is $\big(\mathcal{N}(\mathcal{Z}, d, \frac{\varepsilon}{2}), C \cdot \varepsilon\big)$-uniformly robust for all $\varepsilon > 0$.

The following theorem establishes a connection between the robustness property and generalization performance under an i.i.d. sampling process.

Theorem 23 (Xu and Mannor [2012], Theorem 3)**.** If $\mathcal{S}$ consists of N i.i.d. samples, $\varepsilon > 0$ and $\mathcal{A}$ is (K, ε)-(uniformly) robust, then for all $\delta > 0$, with probability at least $1 - \delta$,

$$|\ell_{\exp}(\mathcal{A}_\mathcal{S}) - \ell_{\mathrm{emp}}(\mathcal{A}_\mathcal{S})| \leq \varepsilon + M\sqrt{\frac{2K \log 2 + 2\log(\frac{1}{\delta})}{N}}.$$

In what follows, we aim to relax the i.i.d. assumption in the above result by (i) introducing possible dependencies between samples, leading to generalization analysis in the inductive case, and (ii) deriving a generalization analysis in the transductive case.

Inductive robustness generalization analysis Independence plays a central role in deriving generalization bounds, particularly in the proof of Theorem 23, which relies on applying the Bretagnolle–Huber–Carol inequality to a multinomial vector drawn from an i.i.d. sample [A. W. van der Vaart, 1996, Proposition A.6.6]. Since this inequality ultimately depends on Hoeffding's inequality, the i.i.d. assumption is essential.

To relax this assumption, we follow Janson [2004], who introduced *dependency graphs*, graph structures in which nodes represent random variables and edges indicate potential dependencies, to extend classical concentration bounds to the dependent setting. We formally define dependency graphs in Appendix K.1 and prove Lemma 28, which is used to derive the following generalization bound.

Theorem 24 (Generalized version of Theorem 1)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space, and let $\mathcal{A}$ be a (K, ε)-uniformly robust learning algorithm on $\mathcal{Z}$. Then for any $\delta > 0$, with probability at least $1 - \delta$, and for all samples $\mathcal{S}$ with dependency graph $G[\mathcal{S}]$, we have:

$$|\ell_{\exp}(\mathcal{A}_\mathcal{S}) - \ell_{\mathrm{emp}}(\mathcal{A}_\mathcal{S})| \leq \varepsilon + M\sqrt{\frac{\chi(G[\mathcal{S}])\big(2(K+1)\log 2 + 2\log\big(\frac{1}{\delta}\big)\big)}{|\mathcal{S}|}},$$

where $\chi(G)$ is the chromatic number of the dependency graph G, $\ell_{\exp}(\mathcal{A}_\mathcal{S})$ and $\ell_{\mathrm{emp}}(\mathcal{A}_\mathcal{S})$ denote the expected and empirical losses, respectively, and $M \in \mathbb{N}$ is an upper bound on the loss function.

Transductive robustness generalization analysis In this section, we derive generalization bounds in the transductive setting using robustness. Unlike the inductive case, where robustness alone suffices, we require that the learning algorithm be stable to small changes in the training set. Intuitively, a transductive algorithm is β-stable if swapping one training and one test input changes the model's predictions by at most β. The formal definition is provided in Appendix K.2[Definition 30].

We now derive generalization bounds for robust and stable transductive learning algorithms. In this setting, Azuma's inequality [Azuma, 1967] replaces Hoeffding's inequality (see Appendix D, Theorem 13), and we extend the Bretagnolle–Huber–Carol inequality to Lemma 32 leading to the following generalization result for the transductive setting.

Theorem 25 (Generalized version of Theorem 2)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space, ℓ be a loss function bounded by M satisfying the conditions of Lemma 31. If $\mathcal{A}$ is a transductive learning algorithm on $\mathcal{Z} = \{z_i\}_{i=1}^{m+u}$ with hypothesis class $\mathcal{H}$ that is (K, ε)-uniformly-robust and satisfies uniform transductive stability $\beta > 0$. Then, for every $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds,

$$|R_m - R_u| \leq 2\varepsilon + \left(\frac{1}{\sqrt{m}} + \frac{1}{\sqrt{u}}\right) \cdot M \cdot K \cdot \sqrt{2(K+1)\log 2 + 2\log\left(\frac{1}{\delta}\right)} + C_\ell \beta,$$

K Missing proofs from Section 4

This appendix provides the proofs of our main results, Theorem 1 and Theorem 2. Specifically, we prove their equivalent formulations, Theorem 24 and Theorem 25 in Appendix J, which are stated directly in terms of the robustness definition of Xu and Mannor [2012] rather than the Lipschitz property. We organize the presentation of the proofs as follows. For Theorem 24, we first establish a concentration inequality for dependent data, similar in spirit to the Bretagnolle–Huber–Carol inequality, stated as Lemma 28, which is then used to prove the theorem. Similarly, for Theorem 25, we prove a concentration inequality adapted to the transductive setting (Lemma 32), which forms the basis of the proof of the main result.

K.1 Dependency graphs and concentration inequalities

We begin with the formal definition of dependency graphs, which model dependencies between random variables.

Definition 26 (Dependency graphs)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space and $(E, \mathcal{E})$ a measurable space. Given $n \in \mathbb{N}$ and random variables $X_i \colon \Omega \to E$, for $i \in [n]$, let G be an undirected graph with $V(G) = [n]$. We say that G is a *dependency graph* of $\boldsymbol{X} = (X_1, \ldots, X_n)$, or that $\boldsymbol{X}$ is G-dependent, if for all disjoint subsets $I, J \subset [n]$, whenever I and J are not adjacent in G (i.e., for all $u \in I, v \in J$ such that $\{u, v\} \notin E(G)$), the collections $\{X_i\}_{i \in I}$ and $\{X_j\}_{j \in J}$ are independent.

This structure allows us to extend concentration inequalities designed initially for independent variables to settings where controlled dependencies exist by penalizing the complexity of the corresponding dependency graph, as shown by the following theorem.

Theorem 27 (Janson [2004, Theorem 2.1])**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space, and let $X_i \colon (\Omega, \mathcal{F}) \to \mathbb{R}$, for $i \in [n]$, be random variables with dependency graph G. Suppose that each X_i takes values in a real interval of length at most $c_i \geq 0$ for all $i \in [n]$. Then, for every $t > 0$,

$$P\left(\sum_{i=1}^{n} X_i - \mathbb{E}\left[\sum_{i=1}^{n} X_i\right] \geq t\right) \leq \exp\left(\frac{-2t^2}{\chi(G)\|\boldsymbol{c}\|_2^2}\right),$$

where $\boldsymbol{c} = (c_1, \ldots, c_n)$, and $\chi(G)$ denotes the chromatic number of G. More specifically, the chromatic number $\chi(G)$ in the bound can be replaced by the fractional chromatic number of G, which provides a tighter bound, as it lower bounds $\chi(G)$; see [Janson, 2004].

We now use Theorem 27 to prove a variant of the Bretagnolle–Huber–Carol inequality for non-independent samples through the following lemma.

Lemma 28. Let $(\Omega, \mathcal{F}, P)$ be a probability space, let $n \in \mathbb{N}$, and let $G \in \mathcal{G}_n$. Suppose that $X_i \colon \Omega \to A \subset \mathbb{R}$ are G-dependent random variables with identical distribution μ, for all $i \in [n]$. Furthermore, assume that there exists $K \in \mathbb{N}$ such that the set A can be partitioned into K disjoint subsets, denoted by $\{C_j\}_{j=1}^K$. Define the random variables

$$Z_j = \sum_{i=1}^n \mathbf{1}_{\{X_i \in C_j\}}, \quad \text{for } j \in [K],$$

where $\mathbf{1}$ denotes the indicator function. Then, for all $t > 0$, the following inequality holds,

$$P\left(\sum_{j=1}^K |Z_j - n\mu(C_j)| \geq 2t\right) \leq 2^{K+1} \exp\left(\frac{-2t^2}{\chi(G)n}\right),$$

where, $\chi(G)$ denotes the chromatic number of the dependency graph G.

Proof. We first observe that

$$\begin{aligned}\sum_{j=1}^K |Z_j - n\mu(C_j)| &= \sum_{j:\, Z_j \geq n\mu(C_j)} (Z_j - n\mu(C_j)) + \sum_{j:\, Z_j < n\mu(C_j)} (n\mu(C_j) - Z_j) \\ &= \underbrace{\max_{S \subset [K]} \sum_{j \in S} (Z_j - n\mu(C_j))}_{A} + \underbrace{\max_{S \subset [K]} \sum_{j \in S} (n\mu(C_j) - Z_j)}_{B} \\ &\leq 2 \max\{A, B\}.\end{aligned}$$

Therefore,

$$\begin{aligned}P\left(\sum_{j=1}^K |Z_j - n\mu(C_j)| \geq 2t\right) &\leq P(\max\{A, B\} \geq t) \\ &= P((A \geq t) \cup (B \geq t)) \\ &\leq P(A \geq t) + P(B \geq t).\end{aligned}$$

Now, note that

$$\begin{aligned}P(A \geq t) &= P\left(\bigcup_{S \subset [K]} \left\{\sum_{j \in S} (Z_j - n\mu(C_j)) \geq t\right\}\right) \\ &\leq \sum_{S \subset [K]} P\left(\sum_{j \in S} (Z_j - n\mu(C_j)) \geq t\right) \\ &= \sum_{S \subset [K]} P\left(\sum_{j \in S} Z_j - n \sum_{j \in S} \mu(C_j) \geq t\right).\end{aligned}$$

Now, observe that $\sum_{j \in S} Z_j = \sum_{i=1}^n Y_i^{(S)}$, where $Y_i^{(S)} = \sum_{j \in S} \mathbf{1}_{X_i \in C_j}$, and $Y_i^{(S)} \in \{0, 1\}$ since $\{C_j\}_{j=1}^K$ are pairwise disjoint as a partition. Note also that $\mathbb{E}[Y_i^{(S)}] = \sum_{j \in S} \mu(C_j)$. Finally, it is easy to see that since X_i are G-dependent, the random variables $Y_i^{(S)}$ are also G-dependent. By Theorem 27, we obtain

$$\begin{aligned}P(A \geq t) &\leq \sum_{S \subset [K]} P\left(\sum_{i=1}^n Y_i^{(S)} - \mathbb{E}\left[\sum_{i=1}^n Y_i^{(S)}\right] \geq t\right) \\ &\leq \sum_{S \subset [K]} \exp\left(\frac{-2t^2}{\chi(G)n}\right) \\ &= 2^K \exp\left(\frac{-2t^2}{\chi(G)n}\right).\end{aligned}$$

Similarly, noting that $\sum_{j\in S} -Z_j = \sum_{i=1}^{n} \widetilde{Y}_i^{(S)}$, where $\widetilde{Y}_i^{(S)} = -\sum_{j\in S} \mathbf{1}_{X_i \in C_j}$, $\widetilde{Y}_i^{(S)} \in \{0, -1\}$, and $\mathbb{E}[\widetilde{Y}_i^{(S)}] = -\sum_{j\in S} \mu(C_j)$, we obtain:

$$
\begin{aligned}
P(B \geq t) &\leq \sum_{S\subset[K]} P\left(\sum_{i=1}^{n} \widetilde{Y}_i^{(S)} - \mathbb{E}\left[\sum_{i=1}^{n} \widetilde{Y}_i^{(S)}\right] \geq t\right) \\
&\leq \sum_{S\subset[K]} \exp\left(\frac{-2t^2}{\chi(G)n}\right) \\
&= 2^K \exp\left(\frac{-2t^2}{\chi(G)n}\right).
\end{aligned}
$$

Hence,

$$
P\left(\sum_{j=1}^{K} |Z_j - n\mu(C_j)| \geq 2t\right) \leq 2^{K+1} \exp\left(\frac{-2t^2}{\chi(G)n}\right).
$$

□

We are now ready to prove the first robustness generalization bound for the inductive setting.

Theorem 29 (Restated, Theorem 24)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space, and let $\mathcal{A}$ be a (K, ε)-uniformly robust learning algorithm on $\mathcal{Z}$. Then, for every $\delta > 0$, with probability at least $1-\delta$ (under P), and for all samples $\mathcal{S}$ with dependency graph $G[\mathcal{S}]$, the following inequality holds:

$$
|\ell_{\exp}(\mathcal{A}_{\mathcal{S}}) - \ell_{\text{emp}}(\mathcal{A}_{\mathcal{S}})| \leq \varepsilon + M\sqrt{\frac{\chi(G[\mathcal{S}])\big(2(K+1)\log 2 + 2\log(\frac{1}{\delta})\big)}{|\mathcal{S}|}},
$$

where $\ell_{\exp}(h_{\mathcal{S}})$ denotes the expected loss, $\ell_{\text{emp}}(h_{\mathcal{S}})$ denotes the empirical loss, $\chi(G[\mathcal{S}])$ is the chromatic number of the dependency graph induced

Proof. Let $\{C_j\}_{j=1}^{K}$ be the partition of $\mathcal{Z}$ by the robustness property, and $N_j = \{s \in \mathcal{S} \mid s \in C_j\}$, for $j \in [K]$, then,

$$
\begin{aligned}
&|\ell_{\exp}(\mathcal{A}_{\mathcal{S}}) - \ell_{\text{emp}}(\mathcal{A}_{\mathcal{S}})| \\
&= \left|\sum_{j=1}^{K} \mathbb{E}(\ell(\mathcal{A}_{\mathcal{S}}, \boldsymbol{z}) \mid \boldsymbol{z} \in C_j)\mu(C_j) - \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} \ell(\mathcal{A}_{\mathcal{S}}, s)\right| \\
&\leq \left|\sum_{j=1}^{K} \mathbb{E}(\ell(\mathcal{A}_{\mathcal{S}}, \boldsymbol{z}) \mid \boldsymbol{z} \in C_j)\frac{|N_j|}{|\mathcal{S}|} - \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} \ell(\mathcal{A}_{\mathcal{S}}, s)\right| \\
&+ \left|\sum_{j=1}^{K} \mathbb{E}(\ell(\mathcal{A}_{\mathcal{S}}, \boldsymbol{z}) \mid \boldsymbol{z} \in C_j)\mu(C_j) - \sum_{j=1}^{K} \mathbb{E}(\ell(\mathcal{A}_{\mathcal{S}}, \boldsymbol{z}) \mid \boldsymbol{z} \in C_j)\frac{|N_j|}{|\mathcal{S}|}\right| \\
&\leq \left|\frac{1}{|\mathcal{S}|}\sum_{j=1}^{K}\sum_{s\in N_j} \max_{\boldsymbol{z}_2\in C_j} |\ell(\mathcal{A}_{\mathcal{S}}, s) - \ell(\mathcal{A}_{\mathcal{S}}, \boldsymbol{z}_2)|\right| + \left|\max_{\boldsymbol{z}\in\mathcal{Z}} |\ell(\mathcal{A}_{\mathcal{S}}, \boldsymbol{z})| \sum_{j=1}^{K}\left|\frac{|N_j|}{|\mathcal{S}|} - \mu(C_j)\right|\right| \\
&\leq \varepsilon + M\sum_{j=1}^{K}\left|\frac{|N_j|}{|\mathcal{S}|} - \mu(C_j)\right| \\
&\leq \varepsilon + M\sqrt{\frac{\chi(G[\mathcal{S}])\big(2(K+1)\log 2 + 2\log(\frac{1}{\delta})\big)}{|\mathcal{S}|}},
\end{aligned}
$$

where in the last inequality we applied Lemma 28. □

K.2 Transductive and concentration inequalities

We now continue with the transductive setting. First, we must formally define the transductive stability property described in Section 4.

Before defining the notion of stability, we introduce the following notation. Given a permutation $\pi \in S_n$ and indices $i, j \in [n]$ with $i \neq j$, we denote by $\pi^{i,j}$ the permutation $\overline{\pi} \in S_n$ obtained by swapping the elements at positions i and j in π. More precisely, $\overline{\pi}(k) = \pi(k)$, for $k \neq i, j$, while $\overline{\pi}(i) = \pi(j)$ and $\overline{\pi}(j) = \pi(i)$. We now formally define the notion of stability following an adaptation of Bousquet and Elisseeff [2002].

Definition 30 (Uniform transductive stability)**.** A transductive learning algorithm $\mathcal{A}$ has uniform transductive stability $\beta > 0$ if, for all $\pi \in S_n$, $i \in [m]$, and $j \in [n] \setminus [m]$, the following holds,

$$\max_{1 \leq k \leq n} |\mathcal{A}_\pi(x_k) - \mathcal{A}_{\pi^{i,j}}(x_k)| \leq \beta.$$

The following lemma shows that if a transductive learning algorithm satisfies uniform stability and the loss function is Lipschitz continuous, then the absolute difference between the expected empirical risk R_m and the expected transductive risk R_u is bounded by a constant that depends on the stability parameter and the Lipschitz constant.

Lemma 31. Let $\mathcal{A}$ be a transductive learning algorithm satisfying uniform transductive stability β, for $\beta > 0$, and $\mathcal{Y} = \{0, 1\}$ or $\mathbb{R}$. Then, for a loss function $\ell \colon \mathcal{H} \times V(G) \times \mathcal{Y} \to \mathbb{R}^+$ that is bounded by $M > 0$ and satisfies the following Lipschitz property. For all $\pi \in S_n, x, x' \in V(G), y \in \mathcal{Y}$,

$$|\ell(\mathcal{A}_\pi, x, y) - \ell(\mathcal{A}_\pi, x', y)| \leq C_\ell |\mathcal{A}_\pi(x) - \mathcal{A}_\pi(x')|,$$

we have that,

$$|\mathbb{E}_\pi[R_m] - \mathbb{E}_\pi[R_u]| \leq C_\ell \cdot \beta.$$

Proof. By Pechyony [2008, Lemma 7], we have

$$|\mathbb{E}_\pi[R_m] - \mathbb{E}_\pi[R_u]| = \left|\mathbb{E}_{\pi,\, i \sim I_1^m,\, j \sim I_{m+1}^u}[\ell(\mathcal{A}_{\pi^{i,j}}, x_i, y_i) - \ell(\mathcal{A}_\pi, x_i, y_i)]\right|,$$

where $i \sim I_1^m$ denotes that i is sampled uniformly from $\{1, \ldots, m\}$, and $j \sim I_{m+1}^u$ denotes that j is sampled uniformly from $\{m+1, \ldots, u\}$.

Applying the Lipschitz continuity of the loss function ℓ (second inequality), and the uniform stability (third inequality), we obtain:

$$\begin{aligned}
&\left|\mathbb{E}_{\pi,\, i \sim I_1^m,\, j \sim I_{m+1}^u}[\ell(\mathcal{A}_{\pi^{i,j}}, x_i, y_i) - \ell(\mathcal{A}_\pi, x_i, y_i)]\right| \\
&\leq \mathbb{E}_{\pi,\, i \sim I_1^m,\, j \sim I_{m+1}^u}[|\ell(\mathcal{A}_{\pi^{i,j}}, x_i, y_i) - \ell(\mathcal{A}_\pi, x_i, y_i)|] \\
&\leq C_\ell \mathbb{E}_{\pi,\, i \sim I_1^m,\, j \sim I_{m+1}^u}[|\mathcal{A}_{\pi^{i,j}}(x_i) - \mathcal{A}_\pi(x_i)|] \\
&\leq C_\ell \mathbb{E}_{\pi,\, i \sim I_1^m,\, j \sim I_{m+1}^u}[\beta] \\
&= C_\ell \beta.
\end{aligned}$$

□

Below, we prove the concentration inequality for the transductive setting that is required to establish our main theorem.

Lemma 32. Let $(\Omega, \mathcal{F}, P)$ be a probability space, and $n, n' \in \mathbb{N}$ with $n < n'$. If $Z = \{z_1, z_2, \ldots, z_{n'}\}$ be an arbitary finite set, $\{C_j\}_{j \in [K]}$ being a partition of Z, $\pi \sim \text{Unif}(S_{n'})$, and

$$X_j = \sum_{i=1}^{n} \mathbf{1}_{\{z_{\pi(i)} \in C_j\}},$$

then the following inequality holds. For all $S \subset [K]$, and for all $t > 0$,

$$P\left(\left|\sum_{j \in S} X_j - \mathbb{E}_\pi\left(\sum_{j \in S} X_j\right)\right| \geq t\right) \leq 2\text{exp}\left(\frac{-t^2}{2|S|^2 \cdot n}\right).$$

Proof. For $S \subset [K]$, let $f_S \colon S_n \to \{0,1\}$ with

$$f_S(\pi) = \sum_{j\in S}\sum_{i=1}^{n} \mathbf{1}_{\{z_{\pi(i)}\in C_j\}}.$$

We define the filtration $\mathcal{F}_i = \cup_{j=0}^{i}\sigma(\pi(j))$, for $i \in [n]$ where, by convention, $\sigma(\pi(0)) := \{\Omega, \emptyset\}$ is the trivial σ-algebra.

Next, we define $(W_i)_{i=0}^{n}$ as the Doob martingale of the random variable $f(\pi)$ with respect to the filtration $\{\mathcal{F}_i\}_{i=1}^{n}$. That is,

$$W_0 = \mathbb{E}_\pi(f_S(\pi)), \quad W_i = \mathbb{E}(f_S(\pi)|\mathcal{F}_i), \quad \text{for } i \in [n].$$

We aim to apply Azuma's inequality (Theorem 13). To do so, we need to bound the differences $|W_k - W_{k-1}|$, for $k \in [n]$. We have,

$$\begin{aligned}
|W_k - W_{k-1}| &= |\mathbb{E}(f(\pi) \mid \mathcal{F}_k) - \mathbb{E}(f(\pi) \mid \mathcal{F}_{k-1})| \\
&= \left|\mathbb{E}\left(\sum_{j\in S}\sum_{i=1}^{n} \mathbf{1}_{\{z_{\pi(i)}\in C_j\}}\middle|\mathcal{F}_k\right) - \mathbb{E}\left(\sum_{j\in S}\sum_{i=1}^{n} \mathbf{1}_{\{z_{\pi(i)}\in C_j\}}\middle|\mathcal{F}_{k-1}\right)\right| \\
&= \left|\sum_{i=k}^{n}\sum_{j\in S}\Big(\mathbb{E}(\mathbf{1}_{\{z_{\pi(i)}\in C_j\}} \mid \mathcal{F}_k) - \mathbb{E}(\mathbf{1}_{\{z_{\pi(i)}\in C_j\}} \mid \mathcal{F}_{k-1})\Big)\right| \\
&\le \sum_{j\in S}\sum_{i=k}^{n} \underbrace{\left|\mathbb{E}(\mathbf{1}_{\{z_{\pi(i)}\in C_j\}} \mid \mathcal{F}_k) - \mathbb{E}(\mathbf{1}_{\{z_{\pi(i)}\in C_j\}}|\mathcal{F}_{k-1})\right|}_{A_{i,j,k}},
\end{aligned}$$

where we have used the tower property, i.e., $\mathbb{E}(X \mid \mathcal{F}) = \mathbb{E}(X)$ if X is $\mathcal{F}$-measurable.

We now show that $A_{i,j,k} \le \frac{1}{n-k}$, for all $k \le n$, $i \in [n] \setminus [k]$, and $j \in S$. Let

$$m_j^{(k)} = \big|\{i \in [k] \mid z_{\pi(i)} \in C_j\}\big|.$$

Since π is a uniformly random permutation, we have:

$$\mathbb{E}_\pi(\mathbf{1}_{\{z_{\pi(i)}\in C_j\}}|\mathcal{F}_{k-1}) = \frac{|C_j| - m_j^{(k-1)}}{n-(k-1)},$$

and

$$\mathbb{E}_\pi(\mathbf{1}_{\{z_{\pi(i)}\in C_j\}}|\mathcal{F}_k) = \frac{|C_j| - m_j^{(k)}}{n-k} = \frac{|C_j| - m_j^{(k-1)} - \mathbf{1}_{\{z_{\pi(k)}\in C_j\}}}{n-k}.$$

If $\mathbf{1}_{\{z_{\pi(k)}\in C_j\}} = 1$, we set $a = |C_j| + m_j^{(k-1)} - 1$ and $b = n-k$, obtaining

$$A_{i,j,k} = \left|\frac{a}{b} - \frac{a+1}{b+1}\right| = \frac{1}{b+1}\left|\frac{b-a}{b}\right| \le \frac{1}{b}.$$

Similarly, if $\mathbf{1}_{\{z_{\pi(k)}\in C_j\}} = 0$, setting $a = |C_j| + m_j^{(k-1)}$ and $b = n-k$ gives

$$A_{i,j,k} = \left|\frac{a}{b} - \frac{a}{b+1}\right| = \frac{1}{b+1}\left|\frac{b-a}{b}\right| \le \frac{1}{(b+1)} \le \frac{1}{b}.$$

Therefore, we conclude that $A_{i,j,k} \le \frac{1}{n-k}$, which implies

$$|W_k - W_{k-1}| \le |S|.$$

Applying Azuma's inequality, we obtain

$$P(|f_S(\pi) - \mathbb{E}_\pi(f_S(\pi))| \ge t) = P\left(\left|\sum_{j\in S} X_j - \mathbb{E}_\pi\left(\sum_{j\in S} X_j\right)\right| \ge t\right) \le 2\exp\left(\frac{-t^2}{2n|S|^2}\right).$$

□

Finally, we restate and prove the main robustness generalization theorem for the transductive setting.

Theorem 33 (Restated, Theorem 25)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space, ℓ be a loss function bounded by M satisfying the conditions of Lemma 31. If $\mathcal{A}$ is a transductive learning algorithm on $\mathcal{Z} = \{z_i\}_{i=1}^{m+u}$ with hypothesis class $\mathcal{H}$ that is (K, ε)-uniformly-robust and satisfies uniform transductive stability $\beta > 0$. Then, for every $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds,

$$\left| \frac{1}{m} \sum_{i=1}^{m} \ell(\mathcal{A}_\pi, x_{\pi(i)}, y_{\pi(i)}) - \frac{1}{u} \sum_{i=m+1}^{m+u} \ell(\mathcal{A}_\pi, x_{\pi(i)}, y_{\pi(i)}) \right| \leq$$

$$2\varepsilon + \left(\frac{1}{\sqrt{m}} + \frac{1}{\sqrt{u}} \right) \cdot M \cdot K \cdot \sqrt{2(K+1)\log 2 + 2\log\left(\frac{1}{\delta}\right)} + C_\ell \beta,$$

where M is an upper bound for the loss function ℓ.

Proof. We begin by considering the absolute difference:

$$\begin{aligned}
&|R_m - R_u| \\
&\leq |R_m - \mathbb{E}_\pi[R_m]| + |R_u - \mathbb{E}_\pi[R_u]| + |\mathbb{E}_\pi[R_m] - \mathbb{E}_\pi[R_u]| \\
&= |R_m - \mathbb{E}_\pi[\ell(\mathcal{A}_\pi, x_{\pi(1)}, y_{\pi(1)})]| + |R_u - \mathbb{E}_\pi[\ell(\mathcal{A}_\pi, x_{\pi(m+1)}, y_{\pi(m+1)})]| + |\mathbb{E}_\pi[R_m] - \mathbb{E}_\pi[R_u]| \\
&\leq |R_m - \mathbb{E}_\pi[\ell(\mathcal{A}_\pi, x_{\pi(1)}, y_{\pi(1)})]| + |R_u - \mathbb{E}_\pi[\ell(\mathcal{A}_\pi, x_{\pi(m+1)}, y_{\pi(m+1)})]| + C_\ell \beta.
\end{aligned}$$

The first inequality follows from the triangle inequality. The equality follows from the symmetry of the learning algorithm (Remark 3). The last inequality is obtained by applying Lemma 31.

Next, we proceed similarly to the proof of Theorem 24, but we employ Lemma 32 instead of Lemma 28 to bound the term $|R_m - \mathbb{E}_\pi[\ell(\mathcal{A}_\pi, x_{\pi(1)}, y_{\pi(1)})]|$. Using the same reasoning, we can bound $|R_u - \mathbb{E}_\pi[\ell(\mathcal{A}_\pi, x_{\pi(m+1)}, y_{\pi(m+1)})]|$.

Defining $N_j^m = \{i \in [m] \mid z_{\pi(i)} \in C_j\}$, we obtain,

$$\begin{aligned}
&|R_m - \mathbb{E}_\pi[\ell(\mathcal{A}_\pi, x_{\pi(1)}, y_{\pi(1)})]| = \left| \sum_{j=1}^{K} \mathbb{E}_\pi(\ell(\mathcal{A}_\pi, z_{\pi(1)}) \mid z_{\pi(1)} \in C_j) P(z_{\pi(1)} \in C_j) - \frac{1}{m} \sum_{i=1}^{m} \ell(\mathcal{A}_\pi, z_{\pi(i)}) \right| \\
&\leq \left| \sum_{j=1}^{K} \mathbb{E}_\pi(\ell(\mathcal{A}_\pi, z_{\pi(1)}) \mid z_{\pi(1)} \in C_j) \frac{|N_j^m|}{m} - \frac{1}{m} \sum_{i=1}^{m} \ell(\mathcal{A}_\pi, z_{\pi(i)}) \right| \\
&\quad + \left| \sum_{j=1}^{K} \mathbb{E}_\pi(\ell(\mathcal{A}_\pi, z_{\pi(1)}) \mid z_{\pi(1)} \in C_j) P(z_{\pi(1)} \in C_j) - \sum_{j=1}^{K} \mathbb{E}_\pi\big(\ell(\mathcal{A}_\pi, z_{\pi(1)}) \mid z_{\pi(1)} \in C_j\big) \frac{|N_j^m|}{m} \right| \\
&\leq \left| \frac{1}{m} \sum_{j=1}^{K} \sum_{z \in N_j^m} \max_{z' \in C_j} |\ell(\mathcal{A}_\pi, z) - \ell(\mathcal{A}_\pi, z')| \right| + \left| \max_{z \in \mathcal{Z}} |\ell(\mathcal{A}_\pi, z)| \sum_{j=1}^{K} \left| \frac{|N_j^m|}{m} - P(z_{\pi(1)} \in C_j) \right| \right| \\
&\leq \varepsilon + M \sum_{j=1}^{K} \left| \frac{|N_j^m|}{m} - \frac{|C_j|}{n} \right|.
\end{aligned}$$

Finally, for $t \geq 0$, we apply Lemma 32 to bound the probability,

$$P\left(\sum_{j=1}^{K} \left| \frac{|N_j^m|}{m} - \frac{|C_j|}{n} \right| \geq t \right).$$

Following the derivations from the proof of Lemma 28, we obtain

$$P\left(\sum_{j=1}^{K} \left| \frac{|N_j^m|}{m} - \frac{|C_j|}{n} \right| \geq t \right) \leq \sum_{S \subset [K]} P\left(\left| \sum_{i=1}^{m} Y_i^{(S)} - \mathbb{E}\left(\sum_{i=1}^{m} Y_i^{(S)} \right) \right| \geq \frac{mt}{2} \right),$$

where

$$Y_i^{(S)} = \sum_{j \in S} \mathbf{1}_{\{z_{\pi(i)} \in C_j\}}.$$

Applying Lemma 28, we obtain

$$P\left(\sum_{j=1}^{K}\left|\frac{|N_j^m|}{m} - \frac{|C_j|}{n}\right| \geq t\right) \leq \sum_{S \subset [K]} 2\exp\left(\frac{-t^2 m}{2|S|^2}\right) \leq 2^{K+1}\exp\left(\frac{-t^2 m}{2K^2}\right).$$

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

$$\sum_{j=1}^{K}\left|\frac{|N_j^m|}{m} - \frac{|C_j|}{n}\right| \leq \frac{1}{\sqrt{m}} \cdot K \cdot \sqrt{2(K+1)\log 2 + 2\log\left(\frac{1}{\delta}\right)}.$$

Following the same reasoning for

$$|R_u - \mathbb{E}_\pi(R_u)|,$$

we derive,

$$|R_m - R_u| \leq 2\varepsilon + \left(\frac{1}{\sqrt{m}} + \frac{1}{\sqrt{u}}\right) \cdot M \cdot K \cdot \sqrt{2(K+1)\log 2 + 2\log\left(\frac{1}{\delta}\right)} + C_\ell \beta.$$

□

L Missing proofs from Section 5

The main theorems in this section are direct consequences of results established earlier. Specifically, Theorem 3 follows immediately by combining Theorem 24 with Theorem 22, and Theorem 4 by combining Theorem 25 with Theorem 22.

Specifically, the Lipschitz constant C appearing in Theorem 3 and Theorem 4 is the same and is given by

$$C = 2\widetilde{L}C_\ell\left(\prod_{t=1}^{L} L_{\varphi_t}\right),$$

where L_{φ_t} denotes the Lipschitz constant of the t-th message-passing layer as defined in Equation (1), C_ℓ the Lipschitz constant of the loss function, and $\widetilde{L}$ is the Lipschitz constant established in Proposition 20.

On assumptions in node prediction bounds Previous works such as Scarselli et al. [2018], Verma and Zhang [2019], Garg et al. [2020] study generalization in node prediction under an inductive setting, but rely on strong and limiting assumptions. Specifically, they assume that graphs decompose into independent substructures, similarly to the computational trees we defined in Appendix G, and that the data distribution is defined over the product space of these trees and their labels. A more restrictive assumption is that these tree-label pairs are sampled independently, even when the trees originate from the same graph, thus ignoring inherent dependencies within the graph structure. In contrast, our framework treats each graph as a relational object, without reducing it to a collection of unrollings. While we relax the i.i.d. assumption, we still capture dependencies within the training set through a mild and natural condition: independence is assumed only across nodes from different graphs. In contrast, dependencies are allowed among nodes within the same graph. This setting better reflects real-world scenarios and provides more realistic assumptions than prior work.

Finally, we state and prove the following corollary, showing that if we restrict our attention to graphs with maximum node degree q, for some $q \in \mathbb{N}$, we can bound the covering number from Theorem 3.

Corollary 34. Let $\mathcal{G}_{d,q}^{(-1,1)}$ be the space of featured graphs with node features in $(-1,1)^d$ and maximum node degree q, and consider the setting from Theorem 3. Then,

$$|\ell_{\text{exp}}(\mathcal{A}) - \ell_{\text{emp}}(\mathcal{A}_\mathcal{S})| \leq 2C\varepsilon + M\sqrt{\frac{D_\mathcal{S}\left((4\log 2)\left(\frac{3}{\varepsilon}\right)^{d\cdot Q} + 2\log 2 + 2\log\left(\frac{1}{\delta}\right)\right)}{N}}, \quad \text{for all } \varepsilon > 0.$$

where $Q = \frac{q^{L+1}-1}{q-1}$, C, and $D_\mathcal{S}$ as previously.

PATTERN	Mixed-1k-r	Mixed-1k-u	Mixed-4k-r	Mixed-4k-u	Mixed-8k-r	Mixed-8k-u	random	uniform
n_{train}	120000	120000	120000	120000	120000	120000	120000	120000
n_{test}	116232	116232	116232	116232	116232	116232	116232	116232
$\mathcal{D}_S$	186	15	183	15	170	15	186	15
Training loss	0.445 $\pm$ 0.01	1.5885 $\pm$ 0.002	0.44 $\pm$ 0.01	1.5806 $\pm$ 0.003	0.43 $\pm$ 0.011	1.579 $\pm$ 0.01	0.2485 $\pm$ 0.0073	1.5739 $\pm$ 0.0123
Test loss	1.7685 $\pm$ 0.005	1.5203 $\pm$ 0.03	1.763 $\pm$ 0.002	1.516 $\pm$ 0.03	1.749 $\pm$ 0.013	1.515 $\pm$0.034	1.8007 $\pm$ 0.0426	1.5114 $\pm$ 0.0152
Gen. Gap	1.3235	0.0682	1.323	0.0646	1.319	0.064	1.552	0.0625
CLUSTER	Mixed-1k-r	Mixed-1k-u	Mixed-4k-r	Mixed-4k-u	Mixed-8k-r	Mixed-8k-u	random	uniform
n_{train}	120000	120000	120000	120000	120000	120000	120000	120000
n_{test}	116633	116633	116633	116633	116633	116633	116633	116633
$\mathcal{D}_S$	177	15	176	15	177	15	190	15
Training loss	0.1244 $\pm$ 0.003	0.4002 $\pm$ 0.01	0.1187 $\pm$ 0.003	0.3806 $\pm$ 0.0161	0.1157 $\pm$ 0.001	0.394 $\pm$ 0.018	0.1741 $\pm$0.0260	0.2578$\pm$0.1293
Test loss	0.3920 $\pm$ 0.01	0.3359 $\pm$ 0.029	0.3869 $\pm$ 0.001	0.3396 $\pm$ 0.024	0.3882 $\pm$ 0.002	0.356 $\pm$ 0.007	0.7964 $\pm$0.3241	0.3579 $\pm$0.0043
Gen. Gap	0.2676	0.0643	0.2682	0.041	0.2725	0.038	0.6223	0.1001

Table 3: Generalization results for different sampling strategies related to **Q2** for inductive node classification on the CLUSTER and PATTERN dataset. Strategy random refers to training nodes sampled from a few graphs (resulting in higher sample dependency), while strategy uniform uses nodes sampled uniformly across many distinct graphs. In addition we use the Strategy mixed with different numbers of distinct graphs and sampling processes. Both strategies use the same number of training and test nodes. $D_{\mathcal{S}}$ denotes the maximum sampled nodes of a single graph in the training set. Further, n_{train} and n_{test} denote the number of train and test nodes.

Proof. The proof is a straightforward application of the bound on the covering number of a d-dimensional unit Euclidean ball with radius $0 < \epsilon < 1$, which is at most $\left(\frac{3}{\epsilon}\right)^d$ (see Handel [2014, Lemma 5.13]). $\square$

M Experimental analysis

This appendix presents the experimental protocol underlying the results and insights discussed in Section 7. The source code of all methods is available in the supplementary material.

Sampling strategies for Q2 In total we consider three different sampling strategies for inductive node classification tasks in **Q2**. Since we want to estimate the difference in generalization capabilities for sampling methods depending on the number of training graphs sampled from we provide methods using the same number of nodes with different distinct training graphs. Throughout the experiments, we fixed the random seed to ensure the same sampling process in each iteration.

First we provide *random* sampling, selecting a subsample of graphs by randomly ordering the graphs and sampling nodes graph-by-graph (except graphs containing test set nodes), exhausting each graph before moving to the next or stopping once the desired number of nodes is reached. This results in a small number of graphs, where training nodes are obtained exclusively, leading to a larger chromatic number $\chi(G[\mathcal{S}])$ (as in Theorem 24) or, equivalently, a larger $D_{\mathcal{S}}$ (following the notation in Theorem 3).

In contrast *uniform* sampling uses the same number of randomly ordered nodes from each graph in the training dataset. This ensures that at least one node is sampled from every training graph and during training each graph is seen at least once. With the selected number of nodes for both datasets we sample multiple nodes from each graph.

As an addition to uniform and random sampling we provide the *mixed* sampling process. This method uses either random or uniform sampling for a specified number of graphs, denoted by r and u in the name respectively. In addition, we conduct our experiments with 1000, 4000 and 8000 distinct graphs seen during training for each dataset. This allows us to set the number of graphs instead of just setting the number of nodes, resulting in a fine grained exploration of influences on generalization errors seen for both datasets

Throughout all sampling strategies, we first choose a subset of graphs from the dataset and uniformly sample n_{test} nodes to form the test set. We then fix the number of training nodes to n_{train}.

Datasets To investigate **Q1** we focus on the transductive setting for node predictions, and we use the datasets Wisconsin, Cornell, Texas [Pei et al., 2020] available as part of the WebKB dataset available at `https://github.com/bingzhewei/geom-gcn` and Cora dataset [Yang et al., 2016, Sen et al., 2008] available under the CC-BY 4.0 license at `https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.datasets.Planetoid.html`. To investigate **Q2** under the inductive setting, we consider Pattern, and Cluster datasets [Dwivedi et al., 2023]. These datasets are available under the MIT license at `https://github.com/graphdeeplearning/benchmarking-gnns`.

Hidden dimension	**16**	**32**	**64**	**128**	**256**	**512**
Calculated bound (n=500)	3.575	4.602	2.71	4.53	4.04	2.578
Generalization Gap (n=500)	0.471	0.461	0.436	0.438	0.424	0.425
Calculated bound (n=1000)	5.27	9.03	4.59	4.67	5.92	4.94
Generalization Gap (n=1000)	0.242	0.213	0.201	0.204	0.20	0.194

Table 4: Ablation study for the size of hidden layers and node prediction bounds on LSP-ER(n,p,cc) graphs with p=,cc=1

Dataset	**(100,0.002,1)**	**(200,0.002,1)**	**(500,0.002,1)**	**(1000,0.002,1)**
Calculated bound	5.88	5.17	4.53	4.66
Generalization Gap	0.56	0.548	0.44	0.21

Table 5: Evaluation of node prediction bounds with p=0.002, cc=1 for LSP-ER(n,p,cc) graphs

We compute the covering number for all transductive datasets. In the case of the inductive datasets, we omit the covering number computation due to its similarity to the computations provided by [Vasileiou et al., 2024a]. Common dataset statistics and properties are in Table 8.

In addition we consider synthetic datasets generated using Erdos-Reyni graphs for **Q3**. We generate the longest shortest path Erdős–Rényi graph LSP-ER(n,p,cc), used for binary node classification, as follows: We first create an Erdős–Rényi (ER) graph with nodes n and edge probability p. In a second step, we connect currently disconnected components in the graph. For this cc denotes the number of randomly sampled links between disconnected components. For example, would mean one link between disconnected components. This process is executed for pairs of graph components. Then, we identify the longest shortest path and assign label 1 to all nodes involved in this path. In the case of multiple longest shortest paths, we regenerate the graph to ensure an unambiguous result. All other nodes are labeled 0.

We chose this generation process to allow some control over how the covering number of the node space varies across datasets. Especially with few suitable small scale datasets available for transductive node-level tasks we aim to generate graphs to determine the behavior of our generalization bounds. Furthermore, with generated data we are able to select a suitable feature space and conduct ablation studies with regard to graph density. Specifically, smaller p (fixed) and larger n produce sparser graphs in which nodes tend to have similar computation trees. This results in smaller distances between nodes and, consequently, tighter generalization bounds. A similar effect is observed when decreasing p for fixed n.

Neural architectures To address **Q1**, we use randomly initialized GIN and SEAL architectures with 3 layers. We further incorporate node features into the GIN architecture as initial inputs to the model. Nonlinearity is introduced via the ReLU function as discussed in our theoretical examination. A complete list of hyperparameters used can be found in Table 14. Note that the slope of the line that upper bounds the observations in Figure 2 and Figure 4 can be used as an upper bound on the Lipschitz constant C in Theorem 4.

In the case of SEAL, we also use GIN as the underlying GNN architecture. We sample one negative link for each link in the dataset, providing an equal amount of positive and negative links. Across all datasets, we use the commonly used data splits, and for SEAL, an 80/10/10 train-valid-test split. Like the node prediction tasks, we use ReLU nonlinearity and sum pooling to compute each link representation. The subgraph sampling is done as proposed in the original SEAL paper [Zhang and Chen, 2018], omitting the target link in its respective subgraph.

For **Q2** we use the same GIN architecture as in **Q1** but trained it for 200 epochs in order to get a suitable generalization error. Furthermore, we use the hyperparameters detailed in Table 10.

In the case of **Q3** we use a randomly initialized GIN architecture with a hidden dimension of 16 or 32 and 3 layers. We incorporate node features into the GIN arcchitecture as initial inputs to the model. Otherwise we use the same GIN architecture as for **Q1**.

Dataset	**(100,0.0005,1)**	**(200,0.0005,1)**	**(300,0.0005,1)**	**(500,0.0005,1)**	**(1000,0.0005,1)**
Calculated bound	6.59	5.65	4.87	4.83	2.65
Generalization Gap	0.581	0.537	0.527	0.416	0.416

Table 6: Evaluation of node prediction bounds with p=0.0005, cc=1 for LSP-ER(n,p,cc) graphs

Dataset	(200,0.0005,1)	(200,0.001,1)	(500,0.0005,1)	(500,0.001,1)
Calculated bound	5.65	6.07	4.83	5.66
Generalization Gap	0.54	0.574	0.416	0.462

Table 7: Node prediction bounds with increasing edge probability for LSP-ER(n,p,cc) graphs

Table 8: Statistics for each dataset considered in Section 7

	Dataset					
	TEXAS	WISCONSIN	CORNELL	CORA	PATTERN	CLUSTER
# Graphs	1	1	1	1	14000	12000
# Avg. nodes	183	251	183	2708	118.9	117.2
# Avg. edges	325	515	298	10556	6098.9	4303.9
# Classes	5	5	5	7	2	6

Hyperparameters For inductive node classification on Pattern and Cluster, we trained for 200 epochs using a modified version of GIN, aligning with Equation (1). In addition, we tuned the learning rate using the set $\{0.01, 0.001, 0.0001\}$, while using the Adam optimizer [Kingma and Ba, 2015]. Across all tasks and models, we used a batch size of 32 and set dropout to 0.1. Further, we do not use learning rate decay across all datasets. We report results on the inductive node classification tasks and the sampling strategies in Table 1. A list of all hyperparameters used can be found in Table 14 and Table 9 for each experiment.

For the correlation experiments, we use a randomly initialized GIN model and a SEAL model using GIN layers. We use 3 layers each to align the computation with possible computations of the generalization bounds. We set the hidden dimension to 16 for Cornell, Texas, and Wisconsin and 32 for Cora, respectively. Since we do not train the models, we omit further training-specific hyperparameters.

In case of **Q3** we use a randomly initialized GIN model with 3 layers to compute the generalization bounds. In accordance with **Q1** we set the hidden dimension of the model to 16 or 32, dropout to 0.1 and do not train the model. Therefore, we omit training specific hyperparameters. In contrast to real world datasets and trained models we use all nodes available for the evaluation of the generalization bound.

Further, we report our experiments' runtime and memory usage in Table 12 and Table 13. We provide a Pytorch Geometric implementation for each model. All our experiments were executed on a system with 12 CPU cores, an Nvidia L40 GPU, and 120GB of memory.

Experimental protocol and model configuration To evaluate **Q1**, we measure whether the perturbed inputs to the generalized distance lead to perturbations in the MPNN outputs. We use the same 80/10/10 train-valid-test split as for the other tasks. We then select nodes or links randomly from the transductive dataset for each dataset. We then compute the (unrolling-based) distances for two sampled nodes or links and compare them to the Euclidean distance of their respective GIN or SEAL outputs. Sample plots for selected nodes showing the correlation between the generalized distance and MPNN outputs are shown in Figure 2, Figure 3, Figure 4, and Figure 5.

Concerning **Q2**, we provide three scenarios for sampling nodes from the graphs in the dataset. First, we consider the case of the train dataset nodes to be sampled from specific graphs. Secondly, we assume uniform node subsampling across all training graphs. Finally we consider the case of both sampling methods with the number of graphs seen during training fixed. We fix the test set for these experiments to a randomly determined train-valid-test split. We report the obtained results in Table 1 and Table 3, which showcase the difference in generalization performance between each sampling method. Node samples denote the fraction of nodes used for the computation.

In order to get generalization bounds for **Q3** we evaluate the perturbed inputs to the generalized distance lead to perturbations in MPNN outputs. Similar to **Q1** we select nodes randomly from the transductive dataset and compute the unrolling based distances and compare them to euclidean distances of MPNN outputs. To compute the generalization bound we compute an estimate of the Lipschitz constant by linearly bounding the computed MPNN outputs depending on the node distance. In a second step we then compute the generalization bound by computing the optimum covering numbers needed for the bound estimate. With the optimal result obtained through searching over possible covering number we further obtain the loss bound and actual generalization gap from the experimental data of our evaluation. We report results in Table 2, Table 5, Table 6 and Table 7. Further we showcase ablation results for different hidden sizes in Appendix M.

Table 9: Hyperparameters used for correlation experiments in Figure 2, Figure 4 with two and three GIN layers.

	Dataset			
	TEXAS	CORNELL	WISCONSIN	CORA
Embedding dim.	16	16	16	32
Hidden dim.	16	16	16	32
Dropout	0.1	0.1	0.1	0.1
Node samples	1	1	1	0.25

Table 10: Hyperparameters used for inductive node classification experiments in **Q2**.

	Dataset	
	PATTERN	CLUSTER
Learning Rate	0.001	0.001
Batch Size	32	32
Embedding dim.	64	64
Hidden dim.	64	64
Epochs	200	200
LR decay	0	0
Gradient norm	1	1
Dropout	0.1	0.1

Table 11: Hyperparameters used for generalization bound computation experiments on the LSP-ER(n,p,cc) datasets. For LSP-ER(500,p,cc) and LSP-ER(1000,p,cc) we used 32 as hidden dimension, otherwise 16.

Dataset	LSP-ER
Batch Size	32
Embedding dim.	64
Hidden dim.	16/32
Dropout	0.1
Node samples	1

Table 12: Runtime and Memory Usage for each experiment in Section 7. The first value denotes the runtime in seconds of each experiment, and the second value the used VRAM in MB. We do not report VRAM used in the correlation tasks as we only do a single forward pass. All results were obtained on a single computing node with an Nvidia L40 GPU and 128GB of RAM.

	Dataset					
	TEXAS	WISCONSIN	CORNELL	CORA	PATTERN	CLUSTER
Correlation (SEAL, L=2,3)	13.32/-	19.56/-	13.76/-	-	-	-
Correlation (Node, L=2,3)	4.32/-	7.42/-	4.45/-	469.02/-	-	-
Inductive Node Uniform	-	-	-	-	199.08/96.71	169.64 / 78.77
Inductive Node Random	-	-	-	-	257.24/88.67	233.11 / 69.46
Inductive Node Mixed-1k-r	-	-	-	-	193.42/96.39	184.94/75.54
Inductive Node Mixed-1k-u	-	-	-	-	219.44/88.67	196.38/68.14
Inductive Node Mixed-4k-r	-	-	-	-	207.73/96.40	177.82/75.53
Inductive Node Mixed-4k-u	-	-	-	-	216.34/88.67	194.35/68.15
Inductive Node Mixed-8k-r	-	-	-	-	179.41/96.40	158.61/75.10
Inductive Node Mixed-8k-u	-	-	-	-	218.84/88.70	200.69/68.14

	Dataset					
	(100,0.001,1)	(100,0.0005,1)	(100,0.002,1)	(200,0.001,1)	(200,0.0005,1)	(200,0.002,1)
Bound Computation (Q3)	18.10/-	15.31/-	14.78/-	33.21/-	36.71/-	61.02/-

	Dataset					
	(500,0.001,1)	(500,0.0005,1)	(500,0.002,1)	(1000,0.001,1)	(1000,0.0005,1)	(1000,0.002,1)
Bound Computation (Q3)	287.01/-	330.64/-	314.74/-	2157.13/-	3122.56/-	1937.44/-

Table 13: Runtime and Memory Usage for each experiment in Section 7 using the LSP-ER datasets. The first value denotes the runtime in seconds of each experiment, and the second value the used VRAM in MB. We do not report VRAM used in the correlation tasks as we only do a single forward pass. All results were obtained on a single computing node with an Nvidia L40 GPU and 128GB of RAM.

Table 14: Hyperparameters used for correlation experiments in Figure 3, Figure 5. One-to-one sampling describes the process of sampling one negative link to each link obtained from the graph.

	Dataset		
	TEXAS	CORNELL	WISCONSIN
Embedding dim.	16	16	16
Hidden dim.	16	16	16
Link sample size	all	all	all
Link sampling	one to one	one to one	one to one
Dropout	0.1	0.1	0.1
Use node features	True	True	True

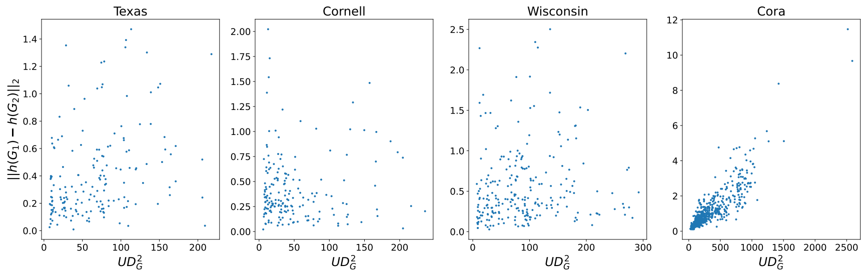

Figure 2: Correlation between GIN-MPNN outputs and the corresponding unrolling distance across real-world datasets for two GIN layers.

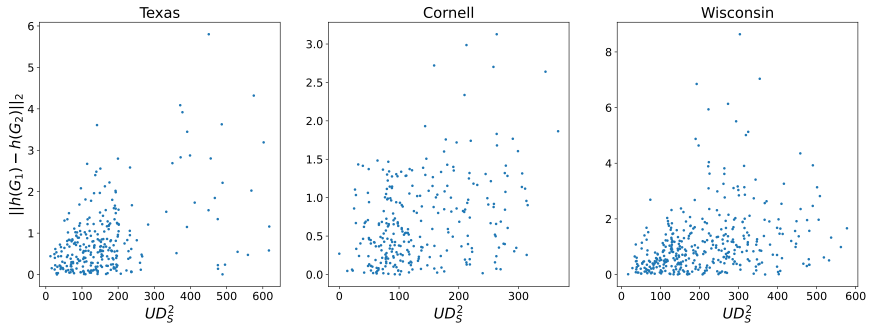

Figure 3: Correlation between SEAL-MPNN outputs and the corresponding unrolling distance across real-world datasets for two GIN layers.

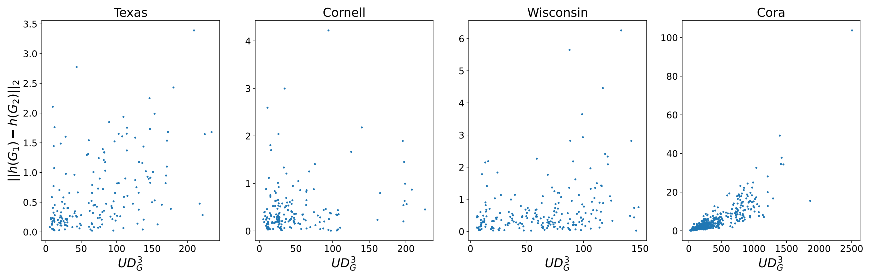

Figure 4: Correlation between GIN-MPNN outputs and the corresponding unrolling distance across real-world datasets for three GIN layers.

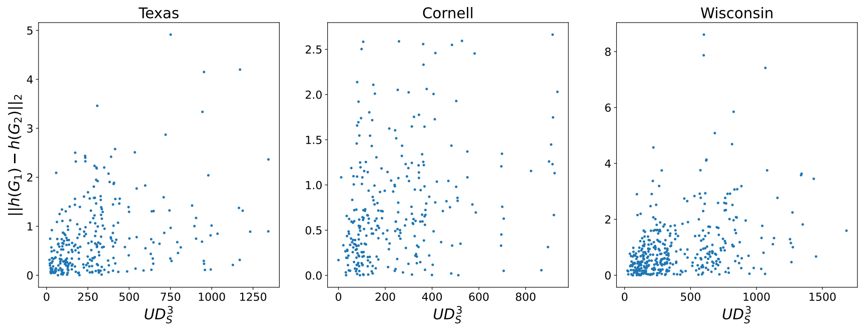

Figure 5: Correlation between SEAL-MPNN outputs and the corresponding unrolling distance across real-world datasets for three GIN layers.