Generalization and Representational Limits of Graph Neural Networks

Vikas K. Garg [1] **Stefanie Jegelka** [1] **Tommi Jaakkola** [1]

Abstract

We address two fundamental questions about graph neural networks (GNNs). First, we prove that several important graph properties, e.g., shortest/longest cycle, diameter, or certain motifs, cannot be computed by GNNs that rely entirely on local information. Such GNNs include the standard message passing models, and more powerful variants that exploit local graph structure (e.g., via relative orientation of messages, or local port ordering) to distinguish neighbors of each node. Our treatment includes a novel graph-theoretic formalism. Second, we provide the first data dependent generalization bounds for message passing GNNs. This analysis explicitly accounts for the local permutation invariance of GNNs. Our bounds are much tighter than existing VC-dimension based guarantees for GNNs, and are comparable to Rademacher bounds for recurrent neural networks.

1. Introduction

Graph neural networks (Scarselli et al., 2009; Gori et al., 2005), in their various incarnations, have emerged as models of choice for embedding graph-structured data from a diverse set of domains, including molecular structures, knowledge graphs, biological networks, social networks, and n-body problems (Duvenaud et al., 2015; Defferrard et al., 2016; Battaglia et al., 2016; Zhang et al., 2018b; Santoro et al., 2018; Yun et al., 2019; Jin et al., 2019).

The working of a graph neural network (GNN) on an input graph, with a feature vector associated with each node, can be outlined as follows. Layer ℓ of the GNN updates the embedding of each node v by aggregating the feature vectors, or node and/or edge embeddings, of v's neighbors from layer $\ell - 1$ via a non-linear transformation, possibly combining this with v's embedding. The exact form of the aggregate and combine steps varies across architectures, and empirical success has been demonstrated for several variants. These include Graph Convolutional Networks (GCN) (Kipf and Welling, 2017), Graph Attention Networks (GAT) (Veličković et al., 2018), Graph Isomorphism Network (GIN) (Xu et al., 2019), and GraphSAGE (Hamilton et al., 2017). GNNs are known to have fundamental connections to message passing (Dai et al., 2016; Gilmer et al., 2017), the Weisfeiler-Leman (WL) graph isomorphism test (Xu et al., 2019; Morris et al., 2019), and local algorithms (Sato et al., 2019; Loukas, 2020).

In this work, we investigate the representational limitations and generalization properties of GNNs. That is, we examine the performance of GNNs from a learning perspective: (a) how well they can discriminate graphs that differ in a specific graph property (represented by assigning different labels to such graphs), and (b) how well they can predict labels, e.g., a graph property, for unseen graphs. Specifically, we focus on classification: (1) a GNN with learnable parameters embeds the nodes of each input graph, (2) the node embeddings are combined into a single graph vector via a *readout* function such as sum, average, or element-wise maximum, and (3) a parameterized classifier either makes a binary prediction on the resulting graph vector (our setting for representation limits), or local binary predictions, one per node, the majority of which determines the graph label (our setting for generalization bounds).

Our contributions. (1) We show that GNNs that generate node embeddings solely based on local information cannot distinguish some simple non-isomorphic graphs. As a result, these GNNs cannot compute important graph properties such as longest or shortest cycle, diameter, etc. This limitation holds for popular models such as GraphSAGE, GCN, GIN, and GAT. Our impossibility results also extend to more powerful variants that provide to each node information about the layout of its neighbors, e.g., via *port numbering*, like CPNGNN (Sato et al., 2019), or geometric information, like DimeNet (Klicpera et al., 2020).

We introduce a novel graph-theoretic formalism for analyzing CPNGNNs, and our constructions provide insights that may facilitate the design of more effective GNNs.

(2) We provide the first data dependent generalization

[1]CSAIL, MIT. Correspondence to: Vikas Garg <vgarg@csail.mit.edu>, Stefanie Jegelka <stefje@mit.edu>, Tommi Jaakkola <tommi@csail.mit.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

bounds for message passing GNNs. Our guarantees are significantly tighter than the VC bounds established by Scarselli et al. (2018) for a class of GNNs. Interestingly, the dependence of our bounds on parameters is comparable to Rademacher bounds for recurrent neural networks (RNNs). Our results also hold for *folding networks* (Hammer, 2001) that operate on tree-structured inputs.

Our generalization analysis specifically accounts for local permutation invariance of the GNN aggregation function. This relies on a specific sum form that extends to aggregating *port-numbered* messages, and therefore opens avenues for analyzing generalization of CPNGNNs.

The first part of the paper is dedicated to analyzing what GNNs cannot learn. The second part investigates how well GNNs learn when they can: we establish new generalization bounds for GNNs. We outline some key proof steps for our results in the main text, and defer the detailed proofs to the Supplementary material.

2. Related Work

GNNs continue to generate much interest from both theoretical and practical perspectives. An important theoretical focus has been on understanding the expressivity of existing architectures, and thereby introducing richer (invariant) models that can generate more nuanced embeddings. But, much less is known about the generalization ability of GNNs. We briefly review some of these works.

Expressivity. Scarselli et al. (2009) extended the universal approximation property of feed-forward networks (FFNs) (Scarselli and Tsoi, 1998) to GNNs using the notion of *unfolding equivalence*. Recurrent neural operations for graphs were introduced with their associated kernel spaces (Lei et al., 2017). Dai et al. (2016) performed a sequence of mappings inspired by mean field and belief propagation procedures from graphical models, and Gilmer et al. (2017) showed that common graph neural net models models may be studied as Message Passing Neural Networks (MPNNs). It is known (Xu et al., 2019) that GNN variants such as GCNs (Kipf and Welling, 2017) and GraphSAGE (Hamilton et al., 2017) are no more discriminative than the Weisfeiler-Leman (WL) test. In order to match the power of the WL test, Xu et al. (2019) also proposed GINs. Showing GNNs are not powerful enough to represent probabilistic logic inference, Zhang et al. (2020) introduced *Express-GNN*.

Among other works, Barceló et al. (2020) proved results in the context of first order logic, and Dehmamy et al. (2019) investigated GCNs through the lens of graph moments underscoring the importance of depth compared to width in learning higher order moments. The inability of some graph kernels to distinguish properties such as planarity has also been established (Kriege et al., 2018; 2020).

Spatial, hierarchical, and higher order GNN variants have also been explored. Notably, Sato et al. (2019) exploited a local port ordering of nodes to introduce the Consistent Port Numbering GNN (CPNGNN), which they proved to be strictly more powerful than WL. They and (Loukas, 2020) also established connections to distributed local algorithms. Higher order generalizations have been studied by (Morris et al., 2019; Murphy et al., 2019; Maron et al., 2019c); in particular, Maron et al. (2019a) introduced models that are more powerful than WL. Hella et al. (2015) investigated models weaker than port numbering. Several other works exploit spatial information to obtain more nuanced embeddings (Ying et al., 2018; You et al., 2019; Ingraham et al., 2019; Klicpera et al., 2020; Chen et al., 2019). Xu et al. (2018) learned locally adaptive structure-aware representations by adaptively aggregating information over extended neighborhoods. Veličković et al. (2018) introduced GATs that obviate specifying the graph structure in advance.

Concurrent to our work, Chen et al. (2020b) established results regarding the ability of MPNNs and Invariant Graph Networks (IGNs) to count substructures in graphs. Subgraph counts and related properties have also been studied in (Arvind et al., 2020). Sato et al. (2020) advocated adding random features to nodes in order to better approximate some problems. We refer the reader to (Sato, 2020) for a survey about expressivity of GNNs.

Invariance. An important consideration in the design of GNNs is their ability to produce output embeddings that are equivariant or permutation-invariant to the input feature vectors. Maron et al. (2019b) constructed permutation-invariant and equivariant linear layers, and showed that their model can approximate any GNN that can be cast as a MPNN in the framework of (Gilmer et al., 2017). Murphy et al. (2019) constructed new permutation-invariant functions for variable-size inputs, and suggested some approximations. Maron et al. (2019c); Keriven and Peyré (2019) proved universality theorems for a specific class of invariant and equivariant networks, respectively.

Generalization. Several works have established generalization guarantees for FFNs (Bartlett et al., 2017; Golowich et al., 2018; Neyshabur et al., 2018; Zhang et al., 2018a) and RNNs (Chen et al., 2020a; Allen-Zhu and Li, 2019). GNNs differ in some key aspects from those models. Unlike RNNs that process sequences, GNNs operate on graph-structured data: sharing of recurrent weights takes place along both the depth and width of a GNN. Unlike FFNs, GNNs deal with irregular local structure. Moreover, at each node, GNNs typically employ permutation-invariant aggregations, in contrast to global permutation invariance (Sokolic et al., 2017). Scarselli et al. (2018) proved VC-dimension bounds for GNNs on a restricted class of graphs

that have their label determined by a single designated node. Verma and Zhang (2019) showed stability bounds for single-layer GCNs in a semi-supervised setting.

Du et al. (2019) introduced *Graph Neural Tangent Kernels* that are equivalent to infinitely wide multi-layer GNNs trained by gradient descent, and established generalization bounds in their large width setting. Xu et al. (2020) studied how the alignment of network architecture and target reasoning tasks affects generalization.

3. Preliminaries

We define the shorthand $[c] = \{1, 2, \ldots, c\}$. For a matrix W, we denote its Frobenius norm by $||W||_F$ and spectral norm by $||W||_2$. We also denote the Euclidean norm of a vector v by $||v||_2$.

In a popular class of GNNs, which we call *Locally Unordered* GNNs (LU-GNNs), the embedding of each node is updated using messages from its neighbors but without using any spatial information (e.g., the relative orientation of the neighbors). This class subsumes variants such as GraphSAGE (Hamilton et al., 2017), GCN (Kipf and Welling, 2017), GIN (Xu et al., 2019), and GAT (Veličković et al., 2018). We can summarize the updated embedding $h_v^{(\ell)}$ for node v at layer ℓ in many LU-GNNs by an aggregation and combine operation:

$$\begin{aligned} \tilde{h}_v^{(\ell-1)} &= \text{AGG}\{h_u^{(\ell-1)} | u \in N(v)\}, \\ h_v^{(\ell)} &= \text{COMBINE}\{h_v^{(\ell-1)}, \tilde{h}_v^{(\ell-1)}\}\,, \end{aligned}$$

where $N(v)$ denotes the set of neighbors of v, and functions AGG and COMBINE are sometimes folded into a single aggregation update. One common implementation, called *mean field embedding* (Dai et al., 2016), uses the input features x_v of node v, in place of $h_v^{(\ell-1)}$ in the COMBINE step above; we will use an instance of this variant for generalization analysis. AGG is typically a permutation-invariant function (e.g., sum).

Recently, two subtle variants have been proposed that exploit local structure to treat the neighbors differently. One of these, CPNGNN (Sato et al., 2019), is based on a *consistent port numbering* that numbers the neighbors of each node v from $1 \ldots degree(v)$. Equivalently, a *port numbering* (or *port ordering*) function p associates with each edge (u, v) a pair of numbers (i, j), $i \in [degree(u)]$ and $j \in [degree(v)]$ such that $p(u, i) = (v, j)$, i.e., u is *connected* to v via *port* i. Thus, u can tell any neighbor from the others based on its ports. We say p is *consistent* if $p(p(u, i)) = (u, i)$ for all (u, i). Multiple consistent orderings are feasible; CPNGNN arbitrarily fixes one before processing the input graph.

When computing node embeddings, the embedding of node v is updated by processing the information from its neighbors as an *ordered* set, ordered by the port numbering, i.e., the aggregation function is generally not permutation invariant. In addition to a neighbor node u's current embedding, v receives the port number that connects u to v.

Another model, *DimeNet* (Klicpera et al., 2020), is a *directional* message passing algorithm introduced in the context of molecular graphs. Specifically, DimeNet embeds atoms via a set of messages (i.e., edge embeddings) and leverages the directional information by transforming messages based on the angle between them. For each node v, the embedding for an incoming message from neighbor u is computed as

$$\begin{aligned} m_{uv}^{(\ell)} &= f_1(m_{uv}^{(\ell-1)}, \tilde{m}_{uv}^{(\ell-1)}), \qquad \text{where} \\ \tilde{m}_{uv}^{(\ell-1)} &= \sum_{w \in N(u) \setminus \{v\}} f_2(m_{wu}^{(\ell-1)}, e^{(uv)}, a^{(wu,uv)}))\,, \end{aligned} \tag{1}$$

and $e^{(uv)}$ is a representation of the distance from u to v, $a^{(wu,uv)}$ combines $\angle wuv$ with the distance from w to u, and f_1 and f_2 are update functions similar to AGG and COMBINE. The node embedding $h_v^{(\ell)}$ is simply the sum of message embeddings $m_{uv}^{(\ell)}$.

For the purposes of this paper, message passing GNNs consist of LU-GNNs, CPNGNN, and DimeNet. We establish representational limits of message passing GNNs with respect to several important graph properties in this paper. Namely, (a) *girth* (length of the shortest cycle), (b) *circumference* (length of the longest cycle), (c) *diameter* (maximum distance, in terms of shortest path, between any pair of nodes in the graph), (d) *radius* (minimum node eccentricity, where eccentricity of a node u is defined as the maximum distance from u to other vertices), (e) *conjoint cycle* (two cycles that share an edge), (f) *total number of cycles*, and (g) k-*clique* (a subgraph of at least $k \geq 3$ vertices such that each vertex in the subgraph is connected by an edge to any other vertex in the subgraph).

We will appeal to the following definition to formalize when GNNs cannot distinguish non-isomorphic graphs, that is, when two distinct graphs are represented with the same embedding regardless of GNN parameters.

Definition 1. For a given graph property P and readout function f, we say that a GNN Q *decides* P, if for any pair of graphs (G_1, G_2) such that G_1 and G_2 differ on P, we have $f(g_Q(G_1)) \neq f(g_Q(G_2))$. Here, $g_Q(G)$ denotes the collection of embeddings of nodes in G when G is provided as input to Q. By extension, a class $\mathcal{Q}$ of GNNs cannot decide P if there does not exist any $Q \in \mathcal{Q}$ that decides P.

We can compute the embedding of any node v using a local *computation tree*, rooted at v and obtained by unrolling

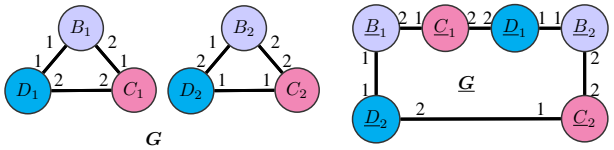


Figure 1: **Construction for Proposition 1**. Graph $\boldsymbol{G}$ consists of two triangles that differ in their ports (shown next to nodes on each edge) but are otherwise identical, whereas $\underline{\boldsymbol{G}}$ consists of a 6-cycle. LU-GNNs do not use ports, and each node treats all its messages equally. Thus, the neighborhood of each node X_1, where $X \in \{B, C, D\}$ in $\boldsymbol{G}$, is indistinguishable from that of $\underline{X}_1$ in $\underline{\boldsymbol{G}}$ (so X_1 and $\underline{X}_1$ have identical embeddings), and similarly X_2 and $\underline{X}_2$ cannot be told apart. So, LU-GNN with permutation-invariant readout fails to separate $\boldsymbol{G}$ and $\underline{\boldsymbol{G}}$. In contrast, CPNGNN can exploit that port 2 of D_2 connects it to B_2, whereas the corresponding node $\underline{D}_2$ connects to $\underline{B}_1$ via port 1.

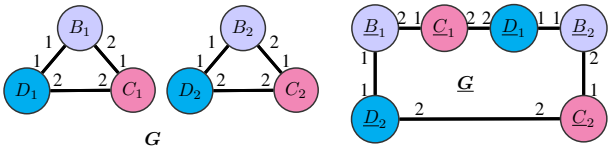


Figure 2: **Construction for Proposition 2**. Graphs $\boldsymbol{G}$ and $\underline{\boldsymbol{G}}$ are as in Fig. 1, but have been assigned a different consistent port numbering. CPNGNN can no longer distinguish the graphs with permutation-invariant readout since each node labeled with X_1 in $\boldsymbol{G}$, where $X \in \{B, C, D\}$ has a corresponding node labeled $\underline{X}_1$ in $\underline{\boldsymbol{G}}$ with identical features and indistinguishable port-numbered neighborhoods (similarly for X_2). Thus, port numbering matters.

the neighborhood aggregations. Specifically, the depth of this tree is L for a GNN with L layers: the children of any node u in the tree are the nodes in $N(u)$. The flow of information is from the leaves all the way up to the root: each leaf is associated with a node feature vector and the embedding of v is simply the result of performing aggregation and combine operations at the root using the embeddings from the subtrees recursively.

4. Representational limits of GNNs

We now sketch novel constructions to illustrate the limits of LU-GNNs, CPNGNNs, and DimeNets. First, we show that in some cases, CPNGNNs can be more discriminative than LU-GNNs, depending on the port numbering. Then, we demonstrate that still, LU-GNNs, CPNGNNs, and DimeNets cannot compute certain graph properties. Our proofs build examples of graphs that (1) differ in important graph properties, but that (2) these models cannot distinguish. As a consequence, these models will not be able to compute such graph properties in general.

To formalize this framework, we introduce a condition of local isomorphism for a pair of graphs. This condition implies that CPNGNNs and LU-GNNs cannot distinguish the two graphs. A similar framework applies to DimeNet. Finally, our insights point to a new GNN variant that leverages additional geometric features to circumvent our constructions for CPNGNNs and DimeNets.

Limitations of LU-GNNs.

Proposition 1. *There exist non-isomorphic graphs that LU-GNNs cannot distinguish, but CPNGNN can distinguish with some consistent port ordering.*

Fig. 1 shows two graphs, $\boldsymbol{G}$ (consisting of two triangles) and $\underline{\boldsymbol{G}}$. Nodes with the same color (or, equivalently, same uppercase symbol without the subscripts and underline) have identical feature vectors. The port numbers for each node are shown next to the node on the respective edges; the numbering is consistent. Moreover, for LU-GNNs, edges on nodes with same color have identical edge feature vectors; for CPNGNNs, edge features are the same if, in addition, the local ports for nodes that have the same color are identical. As explained in Fig. 1, CPNGNN can distinguish between the two graphs by exploiting the port information. However, LU-GNNs do not leverage such information, and fail to find distinct representations.

We note that Theorem 2 in Sato et al. (2019) also establishes the discrepancy in expressivity of CPNGNNs and LU-GNNs, by exploiting a connection between GNNs and local algorithms from distributed computing. In contrast to their approach, our construction in Proposition 1 may also be invoked to argue that DimeNets are strictly more expressive than LU-GNNs.

Limitations of CPNGNNs. Port orderings can help CPNGNNs distinguish graphs that LU-GNNs cannot. But, port orderings are not unique, and not all orderings distinguish the same set of graphs.

Proposition 2. *There exist pairs of non-isomorphic graphs and consistent port numberings p and q such that CPNGNN can distinguish the graphs with p but not q.*

Fig. 2 shows the same pair of graphs $\boldsymbol{G}$ and $\underline{\boldsymbol{G}}$, but with a different ordering. CPNGNN can no longer distinguish the two non-isomorphic graphs with this new ordering. Thus, it may be useful to try multiple random orderings, or even parameterize and learn one along with GNN parameters.

Henceforth, we assume that an ordering is given with the input graph. We now demonstrate the inability of CPNGNNs to decide several graph properties. Toward this goal, note that in Fig. 1 we constructed an explicit bijection

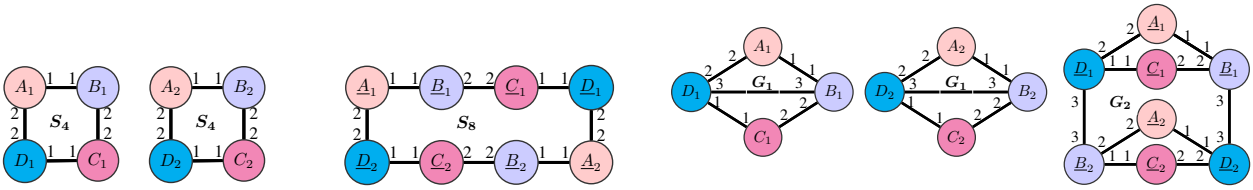


Figure 3: **Constructions for Proposition 4**. The graph with two copies of $\boldsymbol{S_4}$ is indistinguishable from $\boldsymbol{S_8}$ despite having different girth, circumference, diameter, radius, and total number of cycles. This follows since for each $X \in \{A, B, C, D\}$, nodes X_1 and $\underline{X}_1$ have identical feature vectors as well as identical port-ordered neighborhoods (similarly for nodes X_2 and $\underline{X}_2$). Likewise, the graph with two copies of $\boldsymbol{G_1}$, each having a conjoint cycle, cannot be distinguished from $\boldsymbol{G_2}$ as the graphs are port-locally isomorphic. A simple modification extends the result to k-clique (described in the Supplementary). The constructions hold for LU-GNNs as well (by simply ignoring the port numbers). Note that, in contrast, DimeNet is able to distinguish the graphs in these constructions, e.g., using that $\angle A_1 B_1 C_1$ is different from the corresponding $\angle \underline{A}_1 \underline{B}_1 \underline{C}_1$.

between nodes in $\boldsymbol{G}$ and $\underline{\boldsymbol{G}}$ to reason about permutation-invariant readouts. We now introduce a graph-theoretic formalism for CPNGNNs that obviates the need for an explicit bijection and is easier to check.

We define a pair of surjective mappings between two graphs in question, and impose additional conditions that guarantee the existence of a bijection. This bijection implies that corresponding nodes in the graphs receive identical embeddings, and hence both graphs obtain the same set of node embeddings, making them indistinguishable.

The main idea is that a node v_1 in graph G_1 is locally indistinguishable from v_2 in G_2 if (1) the node features agree: $x_{v_1} = x_{v_2}$, and (2) the port-ordered local neighborhoods of v_1 and v_2 cannot be told apart. That is, if port i of v_1 connects to port k of v, then a locality preserving bijection connects the nodes corresponding to images of v_1 and v via the same ports. In the notation here, we include the port numbers (i, j) associated with each edge (u, v) in the edge notation, i.e., $((u, i), (v, j))$.

Definition 2. We say that graph $G_1(V_1, E_1, p)$ *port-covers* $G_2(V_2, E_2, q)$ if the following conditions are satisfied: **(a)** there exists a surjection $f : V_1 \mapsto V_2$ such that $x_v = x_{f(v)}$ for all $v \in V_1$, **(b)** p and q are consistent, and **(c)** for all $v_1 \in V_1$ there exists a local bijection g_{v_1} such that for all $i \in [degree(v_1)]$ and $(v, k) = p(v_i, i)$, we have

$$g_{v_1}(((v_1, i), (v, k))) = (q(f(v), k), q(f(v_1), i)) \ ,$$

such that $q(f(v), k) = (f(v_1), i)$; $q(f(v_1), i) = (f(v), k)$; $((v_1, i), (v, k)) \in E_1$; and $(q(f(v), k), q(f(v_1), i)) \in E_2$. Moreover, we say that $G_1(V_1, E_1, p)$ and $G_2(V_2, E_2, q)$ are *port-locally isomorphic* if they both cover each other.

Definition 2 does not preclude the possibility that f maps multiple nodes in G_1 to the same node in G_2, or the other way round. Hence, the claim that G_1 and G_2 cannot be distinguished by CPNGNN might not hold. Fortunately, the following result comes to our rescue.

Proposition 3. *If $G_1(V_1, E_1, p)$ and $G_2(V_2, E_2, q)$ are port-locally isomorphic, there exists a bijection h that satisfies* **(a)**-**(c)** *in Definition 2 (with h replacing f). As a result, CPNGNNs produce identical embeddings for the corresponding nodes in G_1 and G_2, so CPNGNNs cannot separate G_1 and G_2 with permutation-invariant readout.*

We may characterize port-local isomorphism in alternative ways, e.g., in terms of an identical multiset of port-numbered computation trees (i.e., computation trees with associated port numbers on edges). Our formalism allows us to construct an explicit port-cover bijection from only a pair of surjective mappings, one in each direction (see the proof of Proposition 3 for details). Thus, it provides insight into the role played by locally isomorphic neighborhoods in unraveling a port-cover bijection whenever it exists.

We now proceed to establish that CPNGNNs are limited in that they fail to decide important graph properties. We can invoke conditions of Proposition 3, or define a bijection, to show the following result (see Fig. 3 for our constructions).

Proposition 4. *There exist consistent port orderings such that CPNGNNs with permutation-invariant readout cannot decide several graph properties: girth, circumference, diameter, radius, conjoint cycle, total number of cycles, and k-clique.*

Clearly, these impossibility results apply to LU-GNNs as well (see Fig. 3). However, as described in Fig. 3, our constructions for CPNGNNs do not work for DimeNets. This immediately leads us to the question whether DimeNets are expressive enough to decide the graph properties.

Limitations of DimeNets. Unfortunately, as we show in Fig. 4, there exists an example of two graphs that differ in several of these properties but cannot be distinguished by DimeNets.

Proposition 5. *DimeNet with permutation-invariant readout cannot decide several graph properties: girth, circumference, diameter, radius, or total number of cycles.*

In fact, as we argue in Fig. 4, augmenting DimeNet with port-numbering would still not be sufficient. Therefore, a natural question that arises is whether we can obtain a more expressive model than both CPNGNN and DimeNet. Leveraging insights from our constructions, we now introduce one such variant, *H-DCPN* (short for *Hierarchical Directional Message Passing Consistent Port Numbering Networks*), that generalizes both CPNGNN and DimeNet.

More powerful GNNs. The main idea is to augment DimeNet not just with port ordering, but also additional spatial information. Observe that the construction in Fig. 4 will fail if for each edge (u, v), we additionally model the set of angles α_{wuvz} between planes $\mathcal{P}(w, u, v)$ and $\mathcal{P}(u, v, z)$ due to neighbors w of u and neighbors z of v. Similarly, we could use the distances between these planes. We denote by Φ_{uv} all such features derived from to these planes. Denote by $m_{uv}^{(\ell)}$ the message from neighbor u of v at time ℓ, and by $\underline{m}_{uv}^{(\ell)} = \underline{f}(m_{uv}^{(\ell)}, \Phi_{uv})$ a refined message that encapsulates the effect of geometric features.

We can incorporate salient aspects of CPNGNN as well. Specifically, we first fix a consistent port numbering, as in CPNGNN. Denote the degree of v by $d(v)$. Let $c_v(j)$ be the neighbor of v that connects to port j of v via port $t_{j,v}$, for $j \in [d(v)]$. We suggest to update the embedding of v as

$$h_v^{(\ell)} = f(h_v^{(\ell-1)}, \underline{m}_{c_v(1)v}^{(\ell-1)}, t_{1,v}, \ldots, \underline{m}_{c_v(d(v))v}^{(\ell-1)}, t_{d(v),v}) \ ,$$

where f can potentially take into account the ordering of its arguments. The update resembles CPNGNN when we define $m_{uv}^{(\ell)} = h_u^{(\ell)}$; and DimeNet when f ignores $h_v^{(\ell-1)}$ (and ports) and we define $m_{uv}^{(\ell)}$ using (1) in section 3. H-DCPN derives its additional discriminative power from the features Φ_{uv} encoded in messages $\underline{m}_{uv}^{(\ell)}$. For instance, the nodes labeled A_1, B_1, C_1, D_1 lie on the same plane in $\boldsymbol{G_3}$. In contrast, the plane defined by nodes with labels $\underline{A}_1, \underline{B}_1, \underline{C}_1$ in $\boldsymbol{G_4}$ is orthogonal to that defined by nodes with labels $\underline{D}_2, \underline{A}_1, \underline{B}_1$; thus allowing H-DCPN to distinguish the node labeled A_1 from the node labeled $\underline{A}_1$ (Fig. 4).

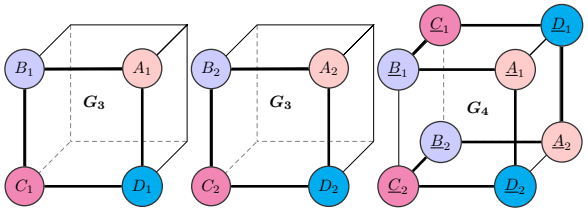


Figure 4: **Construction for Proposition 5**. DimeNet cannot discriminate between $\boldsymbol{G_4}$ and the other graph that consists of two identical copies of $\boldsymbol{G_3}$, since the corresponding local angles and distances are identical in the two graphs. Moreover, since $\boldsymbol{G_3}$ and $\boldsymbol{G_4}$ are obtained by overlaying $\boldsymbol{S_4}$ and $\boldsymbol{S_8}$ (from Fig. 3), respectively on a cube, augmenting DimeNet with the port-numbering scheme from $\boldsymbol{S_4}$ and $\boldsymbol{S_8}$ will still not be sufficient to distinguish the graphs.

5. Generalization bounds for GNNs

Next, we study the generalization ability of GNNs via Rademacher bounds, focusing on binary classification. We generalize the previous results on the complexity of feedforward networks (Bartlett et al., 2017; Neyshabur et al., 2018) and RNNs (Chen et al., 2020a) in mainly three ways. First, we process graphs, unlike sequences in RNNs, or instances restricted to the input layer in feedforward networks. In particular, we show that the complexity of GNNs that combine predictions from individual nodes may be analyzed by focusing on local node-wise computation trees. Second, we share weights across all nodes in these computation trees (i.e., both along the depth and the width of the tree). Third, we model *local* permutation-invariance in the aggregation function at each node in the tree. Curiously, our resulting bounds are comparable to Rademacher bounds for RNNs.

We consider locally permutation invariant GNNs, where in each layer ℓ, the embedding $h_v^\ell \in \mathbb{R}^r$ of node v of a given input graph is updated by aggregating the embeddings of its neighbors, $u \in N(v)$, via an aggregation function $\rho : \mathbb{R}^r \to \mathbb{R}^r$. Different types of updates are possible; we focus on a *mean field* update (Dai et al., 2016; Jin et al., 2018; 2019)

$$h_v^\ell = \phi\big(W_1 x_v + W_2 \rho(\textstyle\sum_{u \in N(v)} g(h_u^{\ell-1}))\big), \qquad (2)$$

where ϕ and g are nonlinear transformations and $x_v \in \mathbb{R}^r$ is the feature vector for v. We assume $\rho(0) = 0$, $||x_v||_2 \leq B_x$ for all v, $||\phi(x)||_\infty \leq b < \infty$ for all $x \in \mathbb{R}^r$, $\phi(0) = 0$, and $g(0) = 0$. One possible choice of ϕ is a squashing function such as tanh. We also assume that ϕ, ρ, and g have Lipschitz constants C_ϕ, C_ρ, and C_g respectively; and that W_1 and W_2 have bounded norms: $||W_1||_2 \leq B_1$, $||W_2||_2 \leq B_2$. The weights W_1, W_2 and functions ϕ, ρ, g are shared across nodes and layers.

The graph label is generated by a readout function that aggregates node embeddings of the final layer L. Here, we assume this function applies a local binary classifier of the form $f_c(h_v^L) = \psi(\beta^\top h_v^L)$ from a family $\mathcal{F}_\beta$ parameterized by β such that $||\beta||_2 \leq B_\beta$, with sigmoid function ψ, to each node representation h_v^L, and then averages the binary predictions of all nodes, i.e., $f(G) = \sum_{v \in V} f_c(h_v^L)$. We predict label 1 if $f(G) > 0.5$, else 0. Such networks implement permutation invariance locally in each neighborhood, and globally when aggregating the node embeddings. This invariance will play an important role in the analysis.

Let $f(G)$ be the output of the entire GNN for input graph G with true label $y \in \{0, 1\}$. Our loss is a margin loss applied

to the difference in probability between true and incorrect label:

$$\tau(f(G), y) = y(2f(G) - 1) + (1 - y)(1 - 2f(G)),$$

with $\tau(f(G), y) < 0$ if and only if there is a classification error. The margin loss is then, with $a = -\tau(f(G), y)$ and indicator function $\mathbf{1}[\cdot]$:

$$\text{loss}_\gamma(a) = \mathbf{1}[a > 0] + (1 + a/\gamma)\mathbf{1}[a \in [-\gamma, 0]]. \quad (3)$$

A standard result in learning theory relates the population risk $\mathbb{P}[\tau(f(G), y) \leq 0]$ to the empirical risk for training examples $\{(G_j, y_j)\}_{j=1}^m$,

$$\hat{\mathcal{R}}_\gamma(f) = \frac{1}{m}\sum_{j=1}^{m} \text{loss}_\gamma(-\tau(f(G_j), y_j)) \quad (4)$$

and to the empirical Rademacher complexity $\hat{\mathcal{R}}_S(\mathcal{J}_\gamma)$ of the class $\mathcal{J}_\gamma$ of functions concatenating the loss with the GNN prediction function f.

Lemma 1 (Mohri et al. (2012))**.** *For any margin $\gamma > 0$, any prediction function f in a class $\mathcal{F}$ and $\mathcal{J}_\gamma \in \{(G, y) \mapsto \text{loss}_\gamma(-\tau(f(G), y)) | f \in \mathcal{F}\}$, given m samples $(G_j, y_j) \sim \mathcal{D}$, with probability $1 - \delta$, the population risk for $\mathcal{D}$ and f is bounded as*

$$\mathbb{P}(\tau(f(G), y) \leq 0) \leq \hat{\mathcal{R}}_\gamma(f) + 2\hat{\mathcal{R}}_\mathcal{G}(\mathcal{J}_\gamma) + 3\sqrt{\tfrac{\log(2/\delta)}{2m}}.$$

Hence, we need to bound the empirical Rademacher complexity $\hat{\mathcal{R}}_\mathcal{G}(\mathcal{J}_\gamma)$ for GNNs. We do this in two steps: (1) we show that it is sufficient to bound the Rademacher complexity of local node-wise computation trees; (2) we bound the complexity for a single tree via recursive spectral bounds, taking into account permutation invariance.

5.1. From Graphs to Trees

We begin by relating the Rademacher complexity of $\mathcal{J}_\gamma$ to the complexity of each node classification. The node embedding h_v^L is equal to a function applied to the local computation tree of depth L, rooted at v, that we obtain when unrolling the L neighborhood aggregations. That is, the tree represents the structured L-hop neighborhood of v, where the children of any node u in the tree are the nodes in $N(u)$. Hence, if t is the tree at v, we may write, with a slight abuse of notation, $f_c(h_v^L) = f_c(t; \Theta)$, where Θ represents the parameters W_1, W_2 of the embedding and β of the node classifier.

With this notation, we rewrite $f(G; \Theta)$ as an expectation over functions applied to trees. Let $T_1, \ldots, T_n$ be the set of all possible computation trees of depth L, and $w_i(G)$ the number of times T_i occurs in G. Then, note that we may write $f(G; \Theta)$ as the sum

$$\sum_{i=1}^{n} \underbrace{\frac{w_i(G)}{\sum_{\ell=1}^n w_\ell(G)}}_{= w_i'(G)} f_c(T_i; \Theta) = \mathbb{E}_{T \sim w'(G)} f_c(T; \Theta). \quad (5)$$

This perspective implies a key insight of our analysis: the complexity of the GNN may be bounded by the complexity of the computation trees.

Proposition 6. *Let $\mathcal{G} = \{G_1, \ldots, G_m\}$ be a set of i.i.d. graphs, and let $\mathcal{T} = \{t_1, \ldots, t_m\}$ be such that $t_j \sim w'(G_j), j \in [m]$. Denote by $\hat{\mathcal{R}}_\mathcal{G}$ and $\hat{\mathcal{R}}_\mathcal{T}$ the empirical Rademacher complexity of GNNs for graphs $\mathcal{G}$ and trees $\mathcal{T}$, respectively. Then $\hat{\mathcal{R}}_\mathcal{G} \leq \mathbb{E}_{t_1,\ldots,t_m} \hat{\mathcal{R}}_\mathcal{T}$.*

Therefore, to apply Lemma 1, it is sufficient to bound the Rademacher complexity of classifying single node-wise computation trees. Before addressing this next step in detail, we state and discuss our main result for this section.

5.2. Generalization Bound for GNNs

We define the *percolation complexity* of our GNNs to be the product $\mathcal{C} \triangleq C_\rho C_g C_\phi B_2$ of Lipschitz constants and weight norm bounds. We now bound $\hat{\mathcal{R}}_\mathcal{T}(\mathcal{J}_\gamma)$, when each tree $t_j \in \mathcal{T}$ has a branching factor (i.e., maximum number of neighbors for any node) at most d, and $\mathcal{J}_\gamma$ maps each (t, y) pair to $\text{loss}_\gamma(-p(f_c(t; \Theta), y))$.

Proposition 7. *The empirical Rademacher complexity of $\mathcal{J}_\gamma$ with respect to $\mathcal{T}$ is*

$$\hat{\mathcal{R}}_\mathcal{T}(\mathcal{J}_\gamma) \leq \frac{4}{\gamma m} + \frac{24 r B_\beta Z}{\gamma\sqrt{m}}\sqrt{3\log Q}\,, \quad \textit{where}$$

$$Q = 24 B_\beta \sqrt{m} \max\{Z, M\sqrt{r}\max\{B_x B_1, \overline{R} B_2\}\},$$

$$M = C_\phi \frac{(\mathcal{C}d)^L - 1}{\mathcal{C}d - 1}\,, \quad Z = C_\phi B_1 B_x + C_\phi B_2 \overline{R}\,,$$

$$\overline{R} \leq C_\rho C_g d \min\left\{b\sqrt{r}, B_1 B_x M\right\}\,.$$

Note that we do not need to prespecify B_1 and B_2: we can simply take these values to be the spectral norm, respectively, of the learned weights W_1 and W_2. Before proceeding with the proof, we discuss some important implications of this result in the wake of Lemma 1 and Proposition 6.

Comparison with RNN. Table 1 summarizes the dependence of our generalization error on the embedding dimension r, branching factor d , depth L and sample size m for different $\mathcal{C}$ up to log factors (denoted by notation $\tilde{\mathcal{O}}$). Importantly, our bounds increase mostly linearly, at most with power 1.5, as a function of r, d and L. Curiously, these dependencies are analogous to bounds for

$\mathcal{C}$	GNN (ours)	RNN (Chen et al. (2020a))
$< 1/d$	$\tilde{\mathcal{O}}\left(\frac{rd}{\sqrt{m}\gamma}\right)$	$\tilde{\mathcal{O}}\left(\frac{r}{\sqrt{m}\gamma}\right)$
$= 1/d$	$\tilde{\mathcal{O}}\left(\frac{rdL}{\sqrt{m}\gamma}\right)$	$\tilde{\mathcal{O}}\left(\frac{rL}{\sqrt{m}\gamma}\right)$
$> 1/d$	$\tilde{\mathcal{O}}\left(\frac{rd\sqrt{rL}}{\sqrt{m}\gamma}\right)$	$\tilde{\mathcal{O}}\left(\frac{r\sqrt{rL}}{\sqrt{m}\gamma}\right)$

Table 1: Our generalization bounds for GNNs are comparable to those for RNNs.

RNNs with sequence of length L (Chen et al., 2020a), when the spectral norm of recurrent weights in RNN plays the role of $\mathcal{C}$. The additional dependence on the branching factor d is due to processing trees in contrast to sequences in RNNs. **C**omparison with VC-bounds for GNNs. Scarselli et al. (2018) established generalization bounds based on VC-dimension. These bounds, for tanh and logistic sigmoid activations, depend with fourth-order on the number of hidden units H, and quadratically on r and the maximum number of nodes N in any input graph. Note that N is at least d, and possibly much larger than d. Since $H = r$ in our setting, this amounts to having the VC-dimension scale as $\mathcal{O}(r^6N^2)$, and consequently, generalization error as $\tilde{\mathcal{O}}(r^3N/\sqrt{m})$. Thus, our generalization bounds are significantly tighter even if $L = \mathcal{O}(r)$.

Local permutation invariance. Previous works focused on permutation-invariance at a global level (Sannai and Imaizumi (2019); Sokolic et al. (2017)). In contrast, GNNs are a composition of permutation-invariant transformations, applied to the neighborhoods of single nodes. We exploit local permutation-invariance via sum-decomposability. In the absence of local permutation-invariance, we would need to address the ordering of messages at each node and the complexity would depend on the worst case ordering.

Extension to other GNN variants. Note that we define C_g and C_ρ with respect to the aggregation function (e.g., unweighted sum or mean) that acts prior to transformation by W_2. This allows us to disentangle the role of shared weights from aggregation. Our analysis easily extends to the setting where messages are weighted (e.g., based on the edge embeddings) prior to aggregation. While we considered message passing with *mean field* updates (Dai et al., 2016), other updates, such as *embedded loopy belief propagation*, may be analyzed similarly in our framework.

Analysis. Our proof proceeds in multiple steps. We first quantify how changing shared weights affects the embedding of the root node of a fixed tree. To do so, we recursively bound the effect on each subtree of the root by the maximum effect across these subtrees. Since both the non-linear activation function and the permutation-invariant aggregation function are Lipschitz-continuous, and the feature vector at the root and the shared weights have bounded norm, the embedding at the root of the tree adapts (i.e., changes gradually in a controlled way) to the embeddings from the subtrees. We then quantify the effect of changing not only the shared weights but also changing the classifier parameters. Since the classifier parameters are chosen from a bounded norm family, we can bound the change in prediction probability. This allows us to use a covering number argument to approximate the predictions, and thus bound the empirical Rademacher complexity via Dudley's entropy integral.

Fix the feature vectors for the computation tree of depth L with degree of each internal node equal to d. Let the feature vector associated with the root (assumed to be at level L) of the tree be given by x_L. We denote the feature vector associated with node j at level $\ell \in [L-1] \triangleq \{1, 2, \ldots, L-1\}$ by $x_{\ell,j}$. Denote the embedding produced by the subtree rooted at node j on level $\ell \in [L-1]$ by $T_{\ell,j}(W_a, W_b)$ when W_a and W_b are the parameters of the model. Consider two sets of parameters $\{W_1, W_2\}$ and $\{W_1', W_2'\}$. We will denote the embedding vector produced by the GNN after processing the entire tree by $T_L(W_1, W_2)$ as a shorthand for $T_{L,1}(W_1, W_2)$. Denote the set of subtrees of node with feature vector x by $C(x)$.

We first quantify the changes in the root embedding when changing the shared weight parameters.

Lemma 2. *The l_2-distance between embedding vectors produced by (W_1, W_2) and (W_1', W_2') after they process the tree from the leaf level to the root can be bounded recursively as*

$$\begin{aligned}\Delta_L &\triangleq ||\, T_L(W_1, W_2) - T_L(W_1', W_2')\,||_2 \\ &\leq C_\phi B_x\, ||(W_1 - W_1')||_2 \;+\; \mathcal{C}d \max_{j \in C(x_L)} \Delta_{L-1,j} \\ &+ C_\phi ||(W_2 - W_2')R(W_1, W_2, x_L)||_2\,,\end{aligned}$$

where

$$R(W_1, W_2, x_L) = \rho\Big(\sum_{j \in C(x_L)} g(T_{L-1,j}(W_1, W_2))\Big)$$

is the permutation-invariant aggregation of the embeddings of the subtrees rooted at level $L-1$ under (W_1, W_2).

We therefore proceed to bounding $||R(W_1, W_2, x_L)||_2$.

Lemma 3.

$$\begin{aligned}&||R(W_1, W_2, x_L)||_2 \\ &\qquad \leq C_\rho C_g d \min\left\{b\sqrt{r}, C_\phi B_1 B_x \frac{(\mathcal{C}d)^L - 1}{\mathcal{C}d - 1}\right\}\end{aligned}$$

Next, we quantify how changing the shared weights and classifier parameters affects the probability of outputting tree label 1.

Lemma 4. *The change in probability Λ_L due to change in parameters from (W_1, W_2, β) to (W_1', W_2', β') is*

$$\begin{aligned}\Lambda_L &= |\psi(\beta^\top T_L(W_1, W_2)) - \psi(\beta'^\top T_L(W_1', W_2'))| \\ &\leq ||\beta - \beta'||_2 Z + B_\beta \Delta_L \,,\end{aligned}$$

where Z is an upper bound on $||T_L(W_1, W_2)||_2$. Moreover, we can bound Δ_L non-recursively:

$$\begin{aligned}\Delta_L \leq\ & M B_x \,||W_1 - W_1'||_2 \\ &+ M||R(W_1, W_2, x_L)||_2 ||W_2 - W_2'||_2 \,.\end{aligned}$$

Lemma 4 allows us to ensure that Λ_L is small via a sufficiently large covering. Specifically, the change in probability Λ_L can be bounded by ϵ, using a covering of size P, where P depends on ϵ. Here, $\log P$ grows as $\mathcal{O}\left(\log\left(1/\epsilon\right)\right)$ for sufficiently small values of ϵ. The stability of prediction translates into good generalization guarantees.

Lemma 5. *The change in probability Λ_L can be bounded by ϵ using a covering of size P, where $\log P$ is at most*

$$3r^2 \log\left(1 + \frac{6B_\beta \max\{Z, M\sqrt{r}\max\{B_x B_1, \overline{R}B_2\}\}}{\epsilon}\right) .$$

Moreover, if the cover parameter (radius)

$$\epsilon < 6B_\beta \max\{Z, M\sqrt{r}\max\{B_x B_1, \overline{R}B_2\}\},$$

then a covering of size P such that $\log P$ is at most

$$3r^2 \log\left(\frac{12B_\beta \max\{Z, M\sqrt{r}\max\{B_x B_1, \overline{R}B_2\}\}}{\epsilon}\right)$$

suffices to ensure $\Lambda_L \leq \epsilon$. Here, $\overline{R} \triangleq ||R(W_1, W_2, x_L)||_2$, and Z, M are as defined in the statement of Proposition 7.

The remaining steps for Proposition 7 are straightforward and deferred to the Supplementary. We now outline some steps for understanding generalization of CPNGNNs.

5.3. Toward a generalization analysis for CPNGNNs

Two parts were integral to our analysis: **(a)** bounding complexity via local computation trees, and **(b)** the sum decomposition property of permutation-invariant functions. We now provide their counterparts for CPNGNNs.

Like before, we start with a vertex v, and unroll the L neighborhood aggregations to obtain a computation tree of depth L, rooted at v. However, now we additionally label each edge in the computation tree with the respective ports of the nodes incident on the edge (enabled by consistent ordering). Thus, we may analyze a input port-numbered graph using its node-wise port-numbered trees.

Since permutation-invariance applies only to multisets of messages, but not port-numbered messages, we cannot express the aggregation in CPNGNNs as in Equation (2). Instead, we provide an injective function for aggregating a collection of port-numbered messages. The function takes a general sum-form that decouples the dependence on each message and its corresponding port number.

Proposition 8. *Assume $\mathcal{X}$ is countable. There exists a function $f : \mathcal{X} \times \mathcal{P} \mapsto \mathcal{R}^n$ such that $h((x_1, p_1), \ldots, (x_{|P|}, p_{|P|})) = \sum_{i \in [|P|]} g(p_i) f(x_i)$ for each port-numbered sequence of (x_i, p_i) pairs, where $P \subset \mathbb{N}$, $X = \{x_1, x_2, \ldots, x_{|P|}\} \subset \mathcal{X}$ is a multiset of bounded size, and p_i are all distinct numbers from $[|P|]$.*

The result in Proposition 8 holds particular significance, since it is known (Hella et al., 2015) that port-numbered messages provide a strictly richer class than sets and multisets. The generalization bound for CPNGNN will be likely worse than the result in Proposition 7, since each port appears as an exponent in our generalized decomposition so the complexity of aggregation grows rapidly in the neighborhood size. We leave a detailed analysis for future work.

Discussion

We discuss some limitations and implications of this work.

Tradeoffs between expressivity and generalization. We introduced H-DCPN primarily to illustrate some ideas that could be used to improve the existing models. In particular, we suggested incorporating geometric information to circumvent some issues with DimeNet. While reasonable for settings where graphs represent actual spatial structures, such models are clearly inapplicable for graphs without any underlying geometric interpretation. Incorporating higher order information via a hypergraph or factor graph might be more natural in some scenarios. Beyond the nature of the graph, a good tradeoff can often be guided by application-specific considerations such as size of the training data, computation budget, and constraints (e.g., on training and inference time). Often, additional expressivity from a complex model might be offset by factors such as computational intractability and lack of generalization.

Theory and practice. Our results shed some light on the observed empirical success and limitations of message passing GNNs. For example, LU-GNNs have been observed to perform suboptimally on molecular graphs, where models that exploit the spatial structure (e.g., DimeNet) perform significantly better (Klicpera et al., 2020). Additional features can avoid some pitfalls of LU-GNNs and have improved performance elsewhere, e.g., Position-aware GNNs (You et al., 2019) and Structured Transformers (Ingraham et al., 2019). Likewise, Proposition 2 elucidates that choosing a good port-numbering can have a crucial impact on the performance of CPNGNNs. Despite addressing complex graphs instead of sequences, generalization bounds for GNNs and RNNs are comparable, so the empirical success of GNNs is not surprising.

Acknowledgments

SJ acknowledges support from NSF awards CAREER 1553284 and 1900933 and from ONR. TJ and VG acknowledge support from ONR and the MIT-IBM collaboration.

References

Z. Allen-Zhu and Y. Li. Can sgd learn recurrent neural networks with provable generalization? In *Neural Information Processing Systems (NeurIPS)*, 2019.

V. Arvind, F. Fuhlbrück, J. Köbler, and O. Verbitsky. On weisfeiler-leman invariance: Subgraph counts and related graph properties. *Journal of Computer and System Sciences*, 113:42 – 59, 2020.

P. Barceló, E. V. Kostylev, M. Monet, J Pérez, J. Reutter, and J.-P. Silva. The logical expressiveness of graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

P. L. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6240–6249, 2017.

P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Neural Information Processing Systems (NIPS)*, pages 4502–4510, 2016.

M. Chen, X. Li, and T. Zhao. On generalization bounds of a family of recurrent neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020a.

Z. Chen, L. Li, and J. Bruna. Supervised community detection with line graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Z. Chen, L. Chen, S. Villar, and J. Bruna. Can graph neural networks count substructures?, 2020b.

H. Dai, B. Dai, and L. Song. Discriminative embeddings of latent variable models for structured data. In *International Conference on Machine Learning (ICML)*, page 2702–2711, 2016.

M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Neural Information Processing Systems (NIPS)*, pages 3844–3852, 2016.

N. Dehmamy, A.-L. Barabasi, and R. Yu. Understanding the representation power of graph neural networks in learning graph topology. In *Neural Information Processing Systems (NeurIPS)*, pages 15387–15397, 2019.

S. S. Du, K. Hou, R. R. Salakhutdinov, B. Poczos, R. Wang, and K. Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Neural Information Processing Systems (NeurIPS)*, 2019.

D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Neural Information Processing Systems (NIPS)*, pages 2224–2232, 2015.

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, pages 1263–1272, 2017.

N. Golowich, A. Rakhlin, and O. Shamir. Size-independent sample complexity of neural networks. In *Conference On Learning Theory (COLT)*, pages 297–299, 2018.

M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 729–734, 2005.

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

B. Hammer. Generalization ability of folding networks. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 13:196–206, 2001.

L. Hella, M. Järvisalo, A. Kuusisto, J. Laurinharju, T. Lempiäinen, K. Luosto, J. Suomela, and J. Virtema. Weak models of distributed computing, with connections to modal logic. *Distributed Computing*, 28(1):31–53, 2015.

J. Ingraham, V. K. Garg, R. Barzilay, and T. Jaakkola. Generative models for graph-based protein design. In *Neural Information Processing Systems (NeurIPS)*, 2019.

W. Jin, R. Barzilay, and T. S. Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning (ICML)*, volume 80, pages 2328–2337, 2018.

W. Jin, K. Yang, R. Barzilay, and T. Jaakkola. Learning multimodal graph-to-graph translation for molecule optimization. In *International Conference on Learning Representations (ICLR)*, 2019.

N. Keriven and G. Peyré. Universal invariant and equivariant graph neural networks. In *Neural Information Processing Systems (NeurIPS)*, pages 7090–7099, 2019.

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

J. Klicpera, J. Groß, and S. Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations (ICLR)*, 2020.

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 (IJCAI)*, pages 2348–2354, 2018.

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

T. Lei, W. Jin, R. Barzilay, and T. Jaakkola. Deriving neural architectures from sequence and graph kernels. In *International Conference on Machine Learning (ICML)*, pages 2024–2033, 2017.

A. Loukas. What graph neural networks cannot learn: depth vs width. *International Conference on Learning Representations (ICLR)*, 2020.

H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman. Provably powerful graph networks. In *Neural Information Processing Systems (NeurIPS)*, pages 2153–2164, 2019a.

H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations (ICLR)*, 2019b.

H. Maron, E. Fetaya, N. Segol, and Y. Lipman. On the universality of invariant networks. In *International Conference on Machine Learning (ICML)*, 2019c.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X, 9780262018258.

C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 4602–4609, 2019.

R. L. Murphy, B. Srinivasan, V. A. Rao, and B. Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In *International Conference on Learning Representations (ICLR)*, 2019.

B. Neyshabur, S. Bhojanapalli, and N. Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

A. Sannai and M. Imaizumi. Improved generalization bound of permutation invariant deep neural networks. *arXiv : 1910.06552*, 2019.

A. Santoro, F. Hill, D. Barrett, A. Morcos, and T. Lillicrap. Measuring abstract reasoning in neural networks. In *International Conference on Machine Learning (ICML)*, pages 4477–4486, 2018.

R. Sato. A survey on the expressive power of graph neural networks, 2020.

R. Sato, M. Yamada, and H. Kashima. Approximation ratios of graph neural networks for combinatorial problems. In *Neural Information Processing Systems (NeurIPS)*, 2019.

R. Sato, M. Yamada, and H. Kashima. Random features strengthen graph neural networks, 2020.

F. Scarselli and A. C. Tsoi. Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural Networks*, 11(1): 15–37, 1998.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks*, 20 (1):81–102, 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.

J. Sokolic, R. Giryes, G. Sapiro, and M. Rodrigues. Generalization Error of Invariant Classifiers. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1094–1103, 2017.

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

S. Verma and Z.-L. Zhang. Stability and generalization of graph convolutional neural networks. In *International Conference on Knowledge Discovery & Data Mining (KDD)*, page 1539–1548, 2019.

K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning (ICML)*, pages 5453–5462, 2018.

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

K. Xu, J. Li, M. Zhang, S. S. Du, K. Kawarabayashi, and S. Jegelka. What can neural networks reason about? In *International Conference on Learning Representations (ICLR)*, 2020.

R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Neural Information Processing Systems (NeurIPS)*, 2018.

J. You, R. Ying, and J. Leskovec. Position-aware graph neural networks. In *International Conference on Machine Learning (ICML)*, pages 7134–7143, 2019.

S. Yun, M. Jeong, R. Kim, J. Kang, and H. Kim. Graph transformer networks. In *Neural Information Processing Systems (NeurIPS)*, pages 11960–11970, 2019.

J. Zhang, Q. Lei, and I. S. Dhillon. Stabilizing gradients for deep neural networks via efficient SVD parameterization. In *International Conference on Machine Learning (ICML)*, pages 5801–5809, 2018a.

M. Zhang, Z. Cui, M. Neumann, and Y. Chen. An end-to-end deep learning architecture for graph classification. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 4438–4445, 2018b.

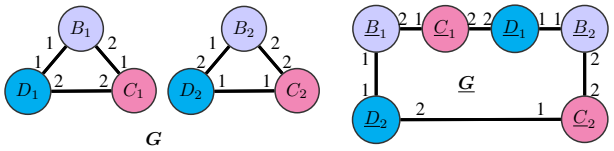
Y. Zhang, X. Chen, Y. Yang, A. Ramamurthy, B. Li, Y. Qi, and L. Song. Efficient probabilistic logic reasoning with graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

A. Supplementary material

We now provide detailed proofs for all our propositions and lemmas.

Proof of Proposition 1

Proof. We show that CPNGNN, using some consistent port ordering, can distinguish some non-isomorphic graphs that LU-GNNs cannot.

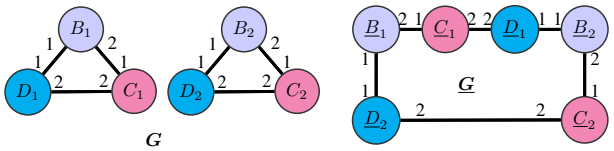


We construct a pair of graphs $\boldsymbol{G}$ and $\underline{\boldsymbol{G}}$ such that $\boldsymbol{G}$ consists of two triangles that differ in port-ordering but are otherwise identical, while $\underline{\boldsymbol{G}}$ (indicated by underlined symbols) consists of a single even-length cycle. The construction ensures that each node labeled with $\boldsymbol{X} \in \{\boldsymbol{B_1}, \boldsymbol{C_1}, \boldsymbol{D_1}, \boldsymbol{B_2}, \boldsymbol{C_2}\}$ in $\boldsymbol{G}$ has the same identical view (i.e., indistinguishable node features, and neighborhood) as the corresponding node labeled $\underline{\boldsymbol{X}}$ in $\underline{\boldsymbol{G}}$. However, $\boldsymbol{D_2}$ and $\underline{\boldsymbol{D_2}}$ have distinguishable neighborhoods due to different port-numbers: e.g., $\boldsymbol{D_2}$ is connected to $\boldsymbol{B_2}$ at port 2, whereas $\underline{\boldsymbol{D_2}}$ is connected to $\underline{\boldsymbol{B_1}}$ at port 1. Likewise, $\boldsymbol{D_2}$ is connected to $\boldsymbol{C_2}$ at port 1, in contrast to $\underline{\boldsymbol{D_2}}$ that is connected to $\underline{\boldsymbol{C_2}}$ at port 2. However, LU-GNN does not incorporate any spatial information such as ports, and fails to tell one graph from the other. □

Note that since $\angle B_1 C_1 D_1$ differs from $\angle \underline{B}_1 \underline{C}_1 \underline{D}_1$, DimeNet can also distinguish between the two graphs.

Proof of Proposition 2

Proof. We now illustrate the importance of choosing a good consistent port numbering. Specifically, we construct a pair of graphs, and two different consistent port numberings p and q such that CPNGNN can distinguish the graphs with p but not q.



We modify the consistent port numbering from the construction of Proposition 1. We consider the same pair of graphs as in the proof of Proposition 1. However, instead of having different numberings for the two components (i.e., triangles) of $\boldsymbol{G}$, we now carry over the ordering from one component to the other. The two components become identical with this modification. For any node labeled $\underline{X}_1$ or $\underline{X}_2$, and any neighbor labeled $\underline{Y}_1$ or $\underline{Y}_2$, $X, Y \in \{B, C, D\}$, we can now simply assign the same respective local ports as the nodes labeled X_1 and Y_1 (or, equivalently, X_2 and Y_2). It is easy to verify that the two graphs become port-locally isomorphic under the new ordering, and thus cannot be separated with any permutation-invariant readout (using Proposition 3).

□

Proof of Proposition 3

Proof. We begin with the following definition.

Definition 3. Two nodes in a graph are *locally indistinguishable* if they have identical feature vectors and identical port-ordered neighborhoods.

In other words, for locally indistinguishable nodes u and v, not only are their neighbors identical but also the respective ports that connect u and v to their identical neighbors are identical.

If the surjection $f : V_1 \to V_2$ in Definition 2 is also injective, then we can simply take $h = f$. Therefore, we focus on the case when f is not injective. We will show that f can be used to inform h. Since f is not injective, there exist $v_1, v_1' \in V_1$ such that $v_1 \neq v_1'$ but $f(v_1) = f(v_1') = v_2$ for some $v_2 \in V_2$. Then, by condition (a) in Definition 2, we immediately get that the feature vector

$$x_{v_1} = x_{f(v_1)} = x_{f(v_1')} = x_{v_1'} \,. \tag{6}$$

Moreover, by other conditions, there is a consistent port bijection from neighborhood of v_1 to that of v_2, and likewise another bijection from neighborhood of v_1' to that of v_2. Therefore, there is a consistent port bijection from neighborhood of v_1 to that of v_1'. Together with (6) and our assumption that $f(v_1) = f(v_1') = v_2$, this implies that v_1 and v_1' are locally indistinguishable. Note that there could be more such nodes that are indistinguishable from v_1 (or v_1'), e.g., when all such nodes map to v_2 as well.

Without loss of generality, let $\mathcal{E}_1(v_1) \subseteq V_1$ denote the equivalence class of all nodes, including v_1, that are indistinguishable from v_1 in graph G_1. Similarly, let $\mathcal{E}_2(v_2) \subseteq V_2$ be the class of nodes indistinguishable from v_2 in G_2. Consider $\ell_1 = |\mathcal{E}_1(v_1)|$ and $\ell_2 = |\mathcal{E}_2(v_2)|$. We claim that $\ell_1 = \ell_2$. Suppose not. Then if $\ell_1 < \ell_2$, we can have h map each node in $\mathcal{E}_1(v_1)$ to a separate node in $\mathcal{E}_2(v_2)$, and use the same mapping as f on the other nodes in V_1. Doing so does not decrease the co-domain of

V_2, and h remains surjective. We are therefore left with $\ell_2 - \ell_1 > 0$ nodes from $\mathcal{E}_2(v_2)$. Therefore, these nodes must have at least one preimage in the set $V_1 - \mathcal{E}_1(v_1)$ since f (and thus h) is a surjection by assumption (**a**) in Definition 1. This is clearly a contradiction since any such preimage must have either a different feature vector, or a non-isomorphic port-consistent neighborhood. By a symmetric argument, using the surjection of map from V_2 to V_1, we conclude that $\ell_1 = \ell_2$. Note that h did not tinker with the nodes that were outside the class $\mathcal{E}_1(v_1)$. Recycling the procedure for other nodes in $V_1 - \mathcal{E}_1(v_1)$ that might map under f to a common image in V_2, we note that h ends up being injective. Since h remains surjective throughout the procedure, we conclude that h is a bijection.

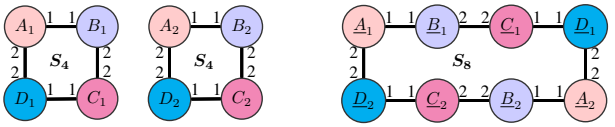
We now prove by induction that the corresponding nodes in port-locally isomorphic graphs have identical embeddings for any CPNGNN. Consider any such GNN with $L+1$ layers parameterized by the sequence $\theta_{1:L+1} \triangleq (\theta_1, \ldots, \theta_L, \theta_{L+1})$. Since there exists a bijection h such that any node $v_1 \in G_1$ has an identical local view (i.e., node features, and port-numbered neighbors) as $v_2 = h(v_1) \in G_1$, the updated embeddings for v_1 and v_2 are identical after the first layer. Assume that these embeddings remain identical after update from each layer $\ell \in \{2, 3, \ldots, L\}$. Since v_1 and v_2 have identical local views and have identical embedding from the Lth layer, the updates for these nodes by the $(L+1)th$ layer are identical. Therefore, v_1 and v_2 have identical embeddings. Since h is a bijection, for every $v \in V_1$ there is a corresponding $h(v) \in V_2$ with the same embedding, and thus both G_1 and G_2 produce the same output with any permutation readout function. Our choice of $\theta_{1:L+1}$ was arbitrary, so the result follows.

□

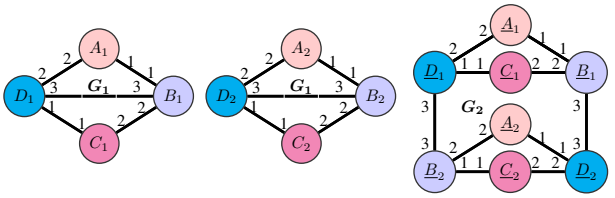
Proof of Proposition 4

Proof. We now show that there exist consistent port orderings such that CPNGNNs with permutation-invariant readout cannot decide several important graph properties: girth, circumference, diameter, radius, conjoint cycle, total number of cycles, and k-clique. The same result also holds for LU-GNNs where nodes do not have access to any consistent port numbering.

We first construct a pair of graphs that have cycles of different length but produce the same output embedding via the readout function. Specifically, we show that CPNGNNs cannot decide a graph having cycles of length n from a cycle of length $2n$. We construct a counterexample for $n = 4$. Our first graph consists of two cycles of length 4 (each denoted by S_4), while the other graph is a cycle of length 8 (denoted by S_8). We associate identical feature vectors with nodes that have the same color, or equivalently, that are marked with the same symbol ignoring the subscripts and the underline. For example, A_1, A_2, $\underline{A}_1$, and $\underline{A}_2$ are all assigned the same feature vector. Moreover, we assign identical edge feature vectors to edges that have the same pair of symbols at the nodes.



Thus, we note that a bijection exists between the two graphs with node X in the first graph corresponding to $\underline{X}$ in the second graph such that both the nodes have identical features and indistinguishable port-ordered neighborhoods. Since, the two graphs have different girth, circumference, diameter, radius, and total number of cycles, it follows from Proposition 3 that CPNGNN cannot decide these properties. Note that the graph with two S_4 cycles is disconnected, and hence its radius (and diameter) is ∞.



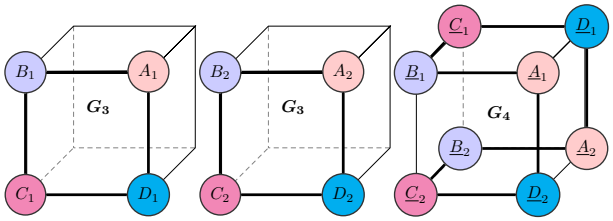
We craft a separate construction for the remaining properties, namely, k-clique and conjoint cycle. The main idea is to replicate the effect of the common edge in the conjoint cycle via two identical components of another graph (that does not have any conjoint cycle) such that the components are cleverly aligned to reproduce the local port-ordered neighborhoods and thus present the same view to each node (see the adjoining figure). Specifically, each conjoint cycle is denoted by G_1, and the other graph that does not have any conjoint cycles by G_2. The graphs, being port-locally isomorphic, are indistinguishable by CPNGNN.

For the k-clique, we simply connect A_1 to C_1, A_2 to C_2, $\underline{A}_1$ to $\underline{C}_1$, and $\underline{A}_2$ to $\underline{C}_2$ via a new port 3 at each of these nodes. Doing so ensures that the new graphs are port-locally isomorphic as well. Adding these edges, we note that, unlike G_2, each conjoint cycle G_1 yields a 4-clique. □

Proof of Proposition 5

Proof. We now demonstrate the representational limits of DimeNets. Specifically, we show two graphs that differ in several graph properties such as girth, circumference, diameter, radius, or total number of cycles. However, these graphs cannot be distinguished by DimeNets.

Note that DimeNet will be able to discriminate S_8 from the graph with cycles S_4 (recall our construction in Proposition 4), since, e.g., $\angle B_1C_1D_1$ in S_4 is different from $\angle \underline{B}_1\underline{C}_1\underline{D}_1$ in S_8. In order to design a failure case for DimeNet, we need to construct a pair of non-isomorphic graphs that have not only identical local pairwise distances but also angles, so that their output embedding is same.



Our idea is to overlay the cycles S_4 and S_8 on a cube (see G_3 and G_4 - the graphs consist of only edges in bold). Doing so does not have any bearing on the graph properties. Since we orient the edges of these cycles along the sides of the cube, the local distances are identical. Moreover, by having $A_1B_1C_1D_1$ and $A_2B_2C_2D_2$ as opposite faces of the cube, we ensure that each angle in G_4 is a right angle, exactly as in G_3. Thus, for each $X \in \{A, B, C, D\}$, nodes X_1, X_2, $\underline{X}_1$, and $\underline{X}_2$ have identical feature vectors and identical local spatial information. Thus, the embeddings for X_1, X_2, $\underline{X}_1$, and $\underline{X}_2$ are identical, and any permutation-invariant readout results in identical output embeddings for the two graphs.

□

Proof of Proposition 6

Proof. We now show that the complexity of the GNN may be bounded by the complexity of the computation trees. In other words, the worst case generalization bound over a set of graphs corresponds to having each graph be a single computation tree. Formally,

$$\begin{aligned}
\hat{\mathcal{R}}_{\mathcal{G}} &\triangleq \mathbb{E}_\sigma \sup_\Theta \sum_{j=1}^{m} \sigma_j f(G_j;\Theta) \\
&= \mathbb{E}_\sigma \sup_\Theta \sum_{j=1}^{m} \sigma_j \mathbb{E}_{T\sim w'(G_j)} f_c(T;\Theta) \\
&\leq \mathbb{E}_\sigma \mathbb{E}_{t_1,..t_m} \sup_\Theta \sum_{j=1}^{m} \sigma_j f_c(t_j;\Theta) \\
&= \mathbb{E}_{t_1,..t_m} \underbrace{\mathbb{E}_\sigma \sup_\Theta \sum_{j=1}^{m} \sigma_j f_c(t_j;\Theta)}_{\hat{\mathcal{R}}_{\mathcal{T}}} ,
\end{aligned}$$

where we invoked Jensen's inequality to swap the expectation with supremum for our inequality (the operation is permissible since sup is a convex function). □

Proof of Lemma 2

Proof. Our objective here is to bound the effect of change in weights from (W_1, W_2) to (W_1', W_2') on the embedding of the root node of our fixed tree (that has depth L). Since non-linear activation and permutation-invariant aggregation are both Lipschitz-continuous functions, and the feature vector at the root x_L and the weights have bounded norm, the embedding at the root of the tree adapts to the embeddings from the subtrees.

Specifically, we note that the l_2-norm of difference of embedding vectors produced by (W_1, W_2) and (W_1', W_2') is

$$\begin{aligned}
\Delta_L &\triangleq ||\,T_L(W_1,W_2) - T_L(W_1',W_2')||_2 \\
&= \Big|\Big|\phi\big(W_1 x_L + W_2\, \rho\big(\underbrace{\sum_{j\in C(x_L)} g(T_{L-1,j}(W_1,W_2))}_{\triangleq\, R(W_1,W_2,x_L)}\big)\big) \\
&- \phi\big(W_1' x_L + W_2' \rho\big(\sum_{j\in C(x_L)} g(T_{L-1,j}(W_1',W_2'))\big)\big)\Big|\Big|_2 \\
&\leq C_\phi \,\big|\big|(W_1 - W_1')x_L\big|\big|_2 \qquad (7) \\
&+ C_\phi ||W_2 R(W_1,W_2,x_L) - W_2' R(W_1',W_2',x_L)||_2 \,.
\end{aligned}$$

Therefore, in order to find an upper bound for Δ_L, we will bound the two terms in the last inequality separately. We first bound the second term using the sum of $||W_2 R(W_1,W_2,x_L) - W_2' R(W_1,W_2,x_L)||_2$ and $||W_2' R(W_1,W_2,x_L) - W_2' R(W_1',W_2',x_L)||_2$. Note that

$$\begin{aligned}
&||W_2' R(W_1,W_2,x_L) - W_2' R(W_1',W_2',x_L)||_2 \\
\leq\; &||W_2'||_2 \;\; ||R(W_1,W_2,x_L) - R(W_1',W_2',x_L)||_2 \,.
\end{aligned} \qquad (8)$$

Since g is C_g-Lipschitz, the branching factor of tree is d, and ρ is C_ρ-Lipschitz, therefore, R is dC_gC_ρ-Lipschitz. We will use this fact to bound (8). Specifically,

$$\begin{aligned}
&||R(W_1,W_2,x_L) - R(W_1',W_2',x_L)||_2 \\
&\leq C_\rho \Big|\Big| \sum_{j\in C(x_L)} \Big(g(T_{L-1,j}(W_1,W_2)) \\
&\qquad\qquad - g(T_{L-1,j}(W_1',W_2'))\Big)\Big|\Big|_2 \\
&\leq C_\rho \sum_{j\in C(x_L)} \Big|\Big|\Big(g(T_{L-1,j}(W_1,W_2)) \\
&\qquad\qquad - g(T_{L-1,j}(W_1',W_2'))\Big)\Big|\Big|_2 \\
&\leq C_\rho C_g \sum_{j\in C(x_L)} \Big|\Big| T_{L-1,j}(W_1,W_2) - T_{L-1,j}(W_1',W_2')\Big|\Big|_2 \\
&= C_\rho C_g \sum_{j\in C(x_L)} \Delta_{L-1,j} \,.
\end{aligned}$$

Using this with $||W_2'||_2 \leq B_2$ in (8), we immediately get

$$\begin{aligned} ||W_2'R(W_1,W_2,x_L) &- W_2'R(W_1',W_2',x_L)||_2 \\ &\leq B_2C_\rho C_g \sum_{j\in C(x_L)} \Delta_{L-1,j} \\ &\leq B_2C_\rho C_g d \max_{j\in C(x_L)} \Delta_{L-1,j}\,. \end{aligned}$$

In other words, we bound the effect on each subtree of the root by the maximum effect across these subtrees. Combining this with $||x_L||_2 \leq B_x$, we note from (7) that

$$\begin{aligned} \Delta_L \;\leq\;& C_\phi B_x\,||(W_1-W_1')||_2 \\ &+ C_\phi B_2 C_\rho C_g d \max_{j\in C(x_L)} \Delta_{L-1,j} \qquad (9) \\ &+ C_\phi ||(W_2-W_2')R(W_1,W_2,x_L)||_2\,. \end{aligned}$$

□

Proof of Lemma 3

Proof. Note from (9) that in order for the change in embedding of the root (due to a small change in weights) to be small, we require that the last term in (9) is small. Toward that goal we bound the norm of permutation-invariant aggregation at the root node. Specifically, we note that

$$\begin{aligned} &||R(W_1,W_2,x_L)||_2 \\ =\;& \left|\left|\rho\Big(\sum_{j\in C(x_L)} g(T_{L-1,j}(W_1,W_2))\Big)\right|\right|_2 \\ =\;& \left|\left|\rho\Big(\sum_{j\in C(x_L)} g(T_{L-1,j}(W_1,W_2))\Big)-\rho(0)\right|\right|_2 \\ \leq\;& C_\rho\left|\left|\sum_{j\in C(x_L)} g(T_{L-1,j}(W_1,W_2))\right|\right|_2 \\ \leq\;& C_\rho \sum_{j\in C(x_L)} \left|\left|g(T_{L-1,j}(W_1,W_2))-g(0)\right|\right|_2 \\ \leq\;& C_\rho C_g \sum_{j\in C(x_L)} \left|\left|T_{L-1,j}(W_1,W_2)\right|\right|_2. \\ \leq\;& C_\rho C_g d \max_{j\in C(x_L)} \left|\left|T_{L-1,j}(W_1,W_2)\right|\right|_2, \qquad (10) \end{aligned}$$

where the norm of the embedding produced by children j of the root using weights W_1 and W_2 is given by

$$\begin{aligned} &\left|\left|T_{L-1,j}(W_1,W_2)\right|\right|_2 \\ =\;& \left|\left|\phi(W_1x_{L-1,j}+W_2R(W_1,W_2,x_{L-1,j}))\right|\right|_2 \\ =\;& \left|\left|\phi(W_1x_{L-1,j}+W_2R(W_1,W_2,x_{L-1,j}))-\phi(0)\right|\right|_2 \\ \leq\;& C_\phi\left|\left|W_1x_{L-1,j}+W_2R(W_1,W_2,x_{L-1,j})\right|\right|_2 \\ \leq\;& C_\phi\left|\left|W_1x_{L-1,j}\right|\right|_2 + C_\phi\left|\left|W_2R(W_1,W_2,x_{L-1,j})\right|\right|_2 \\ \leq\;& C_\phi B_1B_x + C_\phi B_2\left|\left|R(W_1,W_2,x_{L-1,j})\right|\right|_2. \qquad (11) \end{aligned}$$

Also, since $||\phi(x)||_\infty \leq b$ for all $x\in\mathbb{R}^r$ (by our assumption), and $||\phi(x)||_2 \leq \sqrt{r}||\phi(x)||_\infty$, we obtain

$$\left|\left|T_{L-1,j}(W_1,W_2)\right|\right|_2 \;\leq\; b\sqrt{r}\,. \qquad (12)$$

Combining (10) and (11), we get the recursive relationship

$$\begin{aligned} &||R(W_1,W_2,x_L)||_2 \\ \leq\;& C_\rho C_g C_\phi B_1B_x d \\ +\;& C_\rho C_g C_\phi B_2 d \max_{j\in C(x_L)} \left|\left|R(W_1,W_2,x_{L-1,j})\right|\right|_2 \\ \leq\;& C_\rho C_g C_\phi B_1B_x d \sum_{\ell=0}^{L-1} (C_\rho C_g C_\phi B_2 d)^\ell \\ =\;& C_\rho C_g C_\phi B_1B_x d\,\frac{(\mathcal{C}d)^L-1}{\mathcal{C}d-1}\,. \qquad (13) \end{aligned}$$

On the other hand, combining (10) and (11), we get

$$||R(W_1,W_2,x_L)||_2 \;\leq\; bdC_\rho C_g\sqrt{r}\,. \qquad (14)$$

Taken together, (13) and (14) yield $||R(W_1,W_2,x_L)||_2$

$$\leq\; C_\rho C_g d \min\left\{b\sqrt{r}, C_\phi B_1B_x\frac{(\mathcal{C}d)^L-1}{\mathcal{C}d-1}\right\}. \qquad (15)$$

□

Proof of Lemma 4

Proof. Using the results from Lemma 2 and 3, we will simplify the bound on Δ_L, i.e., the change in embedding due to a change in weights. We will then bound the change in probability (that the tree label is 1) Λ_L in terms of Δ_L, when we change not only the weights from (W_1,W_2) to W_1',W_2' but also the local classifier parameters from β to β' (where β and β' are chosen from a bounded norm family). We show these steps below.

Plugging the bound on $\overline{R} \triangleq ||R(W_1, W_2, x_L)||_2$ from Lemma 3 in Lemma 2, we get

$$\begin{aligned}\Delta_L &\leq C_\phi B_x \, ||W_1 - W_1'||_2 \\ &+ \mathcal{C}d \max_{j \in C(x_L)} \Delta_{L-1,j} \\ &+ C_\phi ||W_2 - W_2'||_2 \overline{R} \, .\end{aligned}$$

Expanding the recursion, we note that

$$\Delta_L \leq M B_x \, ||W_1 - W_1'||_2 + M\overline{R}||W_2 - W_2'||_2 \, , \quad (16)$$

where

$$M = C_\phi \frac{(\mathcal{C}d)^L - 1}{\mathcal{C}d - 1} \, . \quad (17)$$

Since $||A||_2 \leq ||A||_F$ for every matrix A, we have

$$\Delta_L \leq M B_x \, ||W_1 - W_1'||_F + M\overline{R}||W_2 - W_2'||_F \, . \quad (18)$$

Now since sigmoid is 1-Lipschitz, we have

$$\begin{aligned}\Lambda_L &= |\psi(\beta^\top T_L(W_1, W_2)) - \psi(\beta'^\top T_L(W_1', W_2'))| \\ &\leq |\beta^\top T_L(W_1, W_2) - \beta'^\top T_L(W_1, W_2)| \\ &+ |\beta'^\top T_L(W_1, W_2) - \beta'^\top T_L(W_1', W_2')| \\ &\leq ||\beta - \beta'||_2 \, ||T_L(W_1, W_2)||_2 + B_\beta \Delta_L \\ &\leq ||\beta - \beta'||_2 \underbrace{(C_\phi B_1 B_x + C_\phi B_2 \overline{R})}_{Z} + B_\beta \Delta_L\end{aligned} \quad (19)$$

using (11) and (15). □

Proof of Lemma 5

Proof. Building on results from Lemmas 2-4, we will now show that the change in probability Λ_L can be bounded by ϵ, using a covering of size P, where P depends on ϵ. Moreover, we show that $\log P$ grows as $\mathcal{O}\left(\log\left(\frac{1}{\epsilon}\right)\right)$ for sufficiently small values of ϵ. That is, we can ensure Λ_L is small by using a small covering.

We begin by noting that we can find a covering $\mathcal{C}\left(\beta, \frac{\epsilon}{3Z_\ell}, ||\cdot||_2\right)$ of size

$$\mathcal{N}\left(\beta, \frac{\epsilon}{3Z_\ell}, ||\cdot||_2\right) \leq \left(1 + \frac{6ZB_\beta}{\epsilon}\right)^r \, .$$

Thus, for any specified ϵ, we can ensure that Λ_L is at most ϵ by finding matrix coverings $\mathcal{C}\left(W_1, \frac{\epsilon}{3MB_xB_\beta}, ||\cdot||_F\right)$ and $\mathcal{C}\left(W_2, \frac{\epsilon}{3M\overline{R}B_\beta}, ||\cdot||_F\right)$. Using Lemma 8 from (Chen et al., 2020a), we obtain the corresponding bounds on their covering number. Specifically,

$$\mathcal{N}\left(W_1, \frac{\epsilon}{3MB_xB_\beta}, ||\cdot||_F\right) \leq \left(1 + \frac{6MB_xB_\beta B_1\sqrt{r}}{\epsilon}\right)^{r^2},$$

$$\mathcal{N}\left(W_2, \frac{\epsilon}{3M\overline{R}B_\beta}, ||\cdot||_F\right) \leq \left(1 + \frac{6M\overline{R}B_\beta B_2\sqrt{r}}{\epsilon}\right)^{r^2}.$$

The product of all the covering numbers is bounded by

$$P = \left(1 + \frac{6B_\beta \max\{Z, M\sqrt{r}\max\{B_xB_1, \overline{R}B_2\}\}}{\epsilon}\right)^{2r^2+r}.$$

Therefore, the class $\mathcal{B}(L, d, r, \beta, B_1, B_2, B_x)$ that maps a tree-structured input to the probability that the corresponding tree label is 1 can be approximated to within ϵ by a covering of size P. Moreover, when

$$\epsilon < 6B_\beta \max\{Z, M\sqrt{r}\max\{B_xB_1, \overline{R}B_2\}\},$$

we obtain that $\log P$ is at most

$$3r^2 \log\left(\frac{12B_\beta \max\{Z, M\sqrt{r}\max\{B_xB_1, \overline{R}B_2\}\}}{\epsilon}\right) .$$

□

Proof of Proposition 7

Proof. We are now ready to prove our generalization bound. Specifically, we invoke a specific form of Dudley's entropy integral to bound the empirical Rademacher complexity $\hat{\mathcal{R}}_\mathcal{T}(\mathcal{J}_\gamma)$ via our result on covering from Lemma 5, where recall that $\mathcal{J}_\gamma$ maps each tree-label pair (t, y) to margin loss $\text{loss}_\gamma(-\tau(f_c(t;\Theta), y))$.

It is straightforward to show that p is 2-Lipschitz in its first argument, and $loss_\gamma$ is $\frac{1}{\gamma}$-Lipschitz. Therefore, we can approximate the class $\mathcal{I}$ that maps (t, y) to $\tau(f_c(t;\Theta), y)$ by finding an $\frac{\epsilon}{2}$-cover of $\mathcal{B}$. Now, note that $\mathcal{I}$ takes values in the interval $[-e, e]$, where

$$e = ||u||_2 \, ||T_L(W_1, W_2)||_2 \leq B_\beta Z \, .$$

Using Lemma A.5. in (Bartlett et al., 2017), we obtain that

$$\hat{\mathcal{R}}_\mathcal{T}(\mathcal{I}) \leq \inf_{\alpha > 0}\left(\frac{4\alpha}{\sqrt{m}} + \frac{12}{m}\int_\alpha^{2e\sqrt{m}} \sqrt{\log \mathcal{N}(\mathcal{I}, \epsilon, ||\cdot||)}d\epsilon\right)$$

where, using Lemma 5, we have

$$\begin{aligned}&\int_\alpha^{2e\sqrt{m}} \sqrt{\log \mathcal{N}(\mathcal{I}, \epsilon, ||\cdot||)}d\epsilon \\ &\leq \int_\alpha^{2e\sqrt{m}} \sqrt{\log \mathcal{N}(\mathcal{B}, \frac{\epsilon}{2}, dist(\cdot,\cdot))}d\epsilon \\ &\leq \int_\alpha^{2e\sqrt{m}} \sqrt{\log U} \quad \leq 2e\sqrt{m}\sqrt{\log U} \quad = 2B_\beta Z\sqrt{m \log U}\end{aligned}$$

with $dist$ being the combination of $||\cdot||_2$ and $||\cdot||_F$ norms used to obtain covering of size P in Lemma 5, and $\log U$ is

$$3r^2 \log\left(\frac{24B_\beta \max\{Z, M\sqrt{r}\max\{B_x B_1, \overline{R}B_2\}\}}{\alpha}\right) .$$

Setting $\alpha = \sqrt{\frac{1}{m}}$, we immediately get

$$\hat{\mathcal{R}}_{\mathcal{T}}(\mathcal{I}) \leq \frac{4}{m} + \frac{24B_\beta Z}{\sqrt{m}}\sqrt{3r^2 \log Q} ,$$

where

$$Q = 24B_\beta\sqrt{m}\max\{Z, M\sqrt{r}\max\{B_x B_1, \overline{R}B_2\}\} .$$

We finally bound the complexity of class $\hat{\mathcal{R}}_{\mathcal{T}}(\mathcal{J}_\gamma)$ by noting that $loss_\gamma$ is $\frac{1}{\gamma}$-Lipschitz, and invoking Talagrand's lemma (Mohri et al., 2012):

$$\hat{\mathcal{R}}_{\mathcal{T}}(\mathcal{J}_\gamma) \leq \frac{\hat{\mathcal{R}}_{\mathcal{T}}(\mathcal{I})}{\gamma} \leq \frac{4}{\gamma m} + \frac{24rB_\beta Z}{\gamma\sqrt{m}}\sqrt{3\log Q} .$$

□

Proof of Proposition 8

Proof. We first convey some intuition. Suppose $|X| < 8$, and we assign a distinct index $z(x) \in \{1, 2, \ldots, 8\}$ to each message $x \in X$. Then, we can map each x to $10^{-z(x)}$, i.e., obtain a decimal expansion which may be viewed as a one-hot vector representation of at most 10 digits. We would reserve a separate block of 10 digits for each port. This would allow us to disentangle the coupling between messages and their corresponding ports. Specifically, since the ports are all distinct, we can *shift* the digits in expansion of x to the right by dividing by 10^p, where p is the port number of x. This allows us to represent each (x, p) pair uniquely.

Formally, since $\mathcal{X}$ is countable, there exists a mapping $Z : \mathcal{X} \mapsto \mathbb{N}$ from $x \in \mathcal{X}$ to natural numbers. Since X has bounded cardinality, we know the existence of some $N \in \mathbb{N}$ such that $|X| < N$ for all X. Define $k = 10^{\lceil \log_{10} N \rceil}$. We define function f in the proposition as $f(x) = k^{-Z(x)}$. We also take function g in proposition to be $g(p) = 10^{-kN(p-1)}$. That is, we express the function h as $h((x_1, p_1), \ldots, (x_{|P|}, p_{|P|})) = \sum_{i=1}^{|P|} g(p_i)f(x_i)$. □