Generalization Analysis of Message Passing Neural Networks on Large Random Graphs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Message passing neural networks (MPNN) have seen a steep rise in popularity since their introduction as generalizations of convolutional neural networks to graph structured data, and are now considered state-of-the-art tools for solving a large variety of graph-focused problems. We study the generalization error of MPNNs in graph classification and regression. We assume that graphs of different classes are sampled from different random graph models. We show that, when training a MPNN on a dataset sampled from such a distribution, the generalization gap increases in the complexity of the MPNN, and decreases, not only with respect to the number of training samples, but also with the average number of nodes in the graphs. This shows how a MPNN with high complexity can generalize from a small dataset of graphs, as long as the graphs are large. The generalization bound is derived from a uniform convergence result, that shows that any MPNN, applied on a graph, approximates the MPNN applied on the geometric model that the graph discretizes.

1 Introduction

A graph is an abstract structure that represents a set of objects along with the connections that exist between those objects. In many important fields, such as chemistry, biology, social networks, or computer graphics, data can be described by graphs. This has led to a tremendous interest in the development of machine learning models for graph-structured data in recent years. A ubiquitous tool for processing such data are graph convolutional neural networks (GCNNs), which extend standard Euclidean convolutional neural networks (CNNs) to graph-structured data.

Most GCNNs used in practice can be described using the general architecture of *Message Passing Neural Networks (MPNNs)*. MPNNs generalize the convolution operator to graph domains by a neighborhood aggregation or message passing scheme. By $\mathbf{f}_i^{(t-1)}$ denoting the feature of node i in layer $t-1$ and $\mathbf{e}_{j,i}$ denoting edge features from node j to i, one layer in a message passing graph neural network is given by

$$\mathbf{f}_i^{(t)} = \Psi^{(t)}\Big(\mathbf{f}_i^{(t-1)}, \mathbf{AGG}\big\{\Phi^{(t)}(\mathbf{f}_i^{(t-1)}, \mathbf{f}_j^{(t-1)}, \mathbf{e}_{j,i})\big\}_{j\in\mathcal{N}(i)}\Big), \tag{1}$$

where $\mathcal{N}(i)$ is the set of nodes connected to node i, $\mathbf{AGG}$ denotes a differentiable and permutation invariant function, e.g., sum, mean, or max, and $\Psi^{(t)}$ and $\Phi^{(t)}$ denote differentiable functions such as MLPs (Multi-Layer Perceptrons) [FL19].

MPNNs have shown state-of-the-art performance in many graph machine learning tasks such as node or graph classification. As such, MPNNs had a tremendous impact to the applied sciences, with promising achievements such as discovering a new class of antibiotics [SYS+20], and has impacted the industry with applications in social media, recommendation systems, and 3D reconstruction, among others (see, e.g., [YHC+18, WHZ+18, WZL+18, MFE+19, FML+19]). The practical success of MPNNs led to a significant boost in research aimed at understanding the theoretical properties of MPNNs. See, e.g., the variational inference point of view of MPNNs [DDS16], and algorithmic alignment of MPNNs with combinatorial algorithms [XHLJ19, MRF+19].

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In this paper we study the generalization capabilities of MPNNs with mean aggregation in a graph classification task. Previous works developed generalization bounds that do not depend on any model of the data, namely, graphs in these works can be generated and labeled in any arbitrary way [STH18, GJJ20, LUZ21]. In this work, we consider a generative model for the graphs which is theoretically powerful and general on the one hand, and allows much tighter generalization bounds on the other hand.

Formally, we are given pairs of graphs and graph signals $\mathbf{x} = (G, \mathbf{f})$ and a target output $\mathbf{y}$, where $(\mathbf{x}, \mathbf{y})$ are jointly drawn from a distribution $\mu_{\mathcal{G}}(\mathbf{x}, \mathbf{y})$. The goal is to learn a MPNN Θ that approximates $\mathbf{y}$ by $\Theta(\mathbf{x})$. For this, one uses a loss function $\mathcal{L}$, which measures the discrepancy between the true label $\mathbf{y}$ and the output of the MPNN $\Theta(\mathbf{x})$. The aim of a machine learning algorithm is to minimize the statistical loss (also called expected loss)

$$R_{exp}(\Theta) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mu_{\mathcal{G}}}\Big[\mathcal{L}(\Theta(\mathbf{x}), \mathbf{y})\Big].$$

In (data-driven) machine learning one has only access to a training set instead of knowing the distribution $\mu_{\mathcal{G}}$. Namely, we consider a multi-graph setting, where the training set $\mathcal{T} = (\mathbf{x}^i = (G^i, \mathbf{f}^i), \mathbf{y}^i)_{i=1}^m$ is a collection of m samples drawn i.i.d. from the distribution $\mu_{\mathcal{G}}(\mathbf{x}, \mathbf{y})$. Then, instead of minimizing the statistical loss, one minimizes the empirical loss, given by

$$R_{\text{emp}}(\Theta) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\Theta(\mathbf{x}^i), \mathbf{y}^i).$$

The optimized MPNN then depends on the dataset, and is hence denoted by $\Theta_{\mathcal{T}}$. The *generalization error* is defined to be

$$GE(\Theta_{\mathcal{T}}) = |R_{\text{exp}}(\Theta_{\mathcal{T}}) - R_{\text{emp}}(\Theta_{\mathcal{T}})|. \tag{2}$$

One then usually bounds (2) by the *uniform generalization error*

$$GE = \sup_{\Theta} |R_{\text{exp}}(\Theta) - R_{\text{emp}}(\Theta)|, \tag{3}$$

where the supremum is taken over some space of MPNNs. Bounds of GE typically take the form $GE^2 \leq \frac{C}{m} q(N)$, where C is a constant that describes the complexity of the model class (e.g., number of parameters), m is the size of the training set, and $q(N)$ is a constant that depends on the (average) size of the graphs. For such bounds, see, e.g., VC-dimension based bounds [STH18], Rademacher complexity based bounds [GJJ20], and PAC-Bayesian based bounds [LUZ21].

While in previous bounds from the literature $q(N)$ either increases in N or in the average degree, in this paper we develop a generalization bound that decays in the average number of nodes N. The idea is to treat the nodes of each graph as randomly sampled from some random graph model. In this point of view, not only the different graphs $\mathbf{x}^i$ are seen as random samples, but the union of all nodes of all graphs comprise together the random samples of the empirical loss. In the spirit of Monte Carlo theory, such a point of view should lead to a decay of the error between the empirical and statistical losses as N increases. As opposed to graphs, nodes cannot be seen as independent, due to the correlations entailed by the graph structure. Hence, our analysis focuses on developing Monte Carlo error bounds in a correlated nodes regime.

Since in our approach we model graphs as randomly sampled from underlying continuous models, we define the application of message passing neural networks, not only on graphs, but also on the underlying space from which graphs are sampled. We then formulate and prove the following convergence result, that we write here informally. Let $\mathbf{x} = (G, \mathbf{f})$ be drawn from the model χ, then with high probability, we have for all MPNNs Θ

$$\|\Theta(\mathbf{x}) - \Theta(\chi)\| = O(N^{-\alpha}),$$

where N is the number of nodes in $\mathbf{x}$ and $\alpha > 0$. Based on this convergence result, we are able prove a generalization bound that decays in N.

1.1 Validity of the Proposed Model

The random graph models in our work are graphons [Lov12] with associated graphon signals (see Definition 2.3). The main assumption in our analysis is that graphs that are sampled from the same

graphon belong to the same class. While this may seem like a limitation, it is actually a very mild and reasonable assumption. It is well known that equivalence classes of isomorphic graphs can be characterized by homomorphism densities [Lov67]. Namely, given two graphs G_1, G_2, if (and only if) for every simple graph F the number of homomorphisms from F to G_1 is equal to the number of homomorphisms from F to G_2, then G_1 is isomorphic to G_2. Graphon analysis relaxes this observation to a continuous similarity measure. A sequence of graphs $\{G_j\}_{j\in\mathbb{N}}$ is said to converge in the graphon sense, if for every simple graph F the homomorphism densities of F in the graphs $\{G_j\}_{j\in\mathbb{N}}$ converge to some value. Graphs from such a sequence can be thought of as being similar in some sense which relaxes the combinatorial notion of graph isomorphism. Moreover, for each such converging sequence, there is a unique (up to some symmetry) limit object, called a graphon. This graphon is also seen as a generative model for graphs in the respective sequence, where graphs are generated by randomly sampling the graphon (see Definition 2.3). Now, since it is well known that MPNNs cannot distinguish between isomorphic graphs, it is also unreasonable to expect them to separate two graphs that are sampled from the same graphon. We hence assume that two graphs that are sampled from the same graphon belong to the same class (but not necessarily vice versa). This assumption allows us to derive a generalization bound that is much tighter than previously proposed bounds (see Figure 1 for comparison).

From a practical stance, our graphon assumption is reasonable since many graph models are special cases of graphons, like Erdős–Rényi, stochastic block model, and random geometric graphs [Pen03]. Moreover, the decoder of a graph variational autoencoder [KW16] can be seen as a graphon.

1.2 Related Work

In this subsection we briefly survey different approaches for studying the convergence and generalization capabilities of GCNNs that were introduced in previous contributions. We give a comparison with our results in Section 3.

In [LHB+21], the authors introduce the notion of GCNN transferability – the ability to transfer a GCNN between different graphs, which is closely related to generalization. For example, [LIK19, GBR20, KTD21] show that the output of spectral-based GCNNs is linearly stable with respect to perturbations of the input graphs. [LHB+21] prove that spectral-based methods are transferable under graphs and graph signals that are sampled from the same latent space. [KBV20, RGR21, RWR21, MLK21] show that spectral-based GCNNs are transferable under graphs that approximate the same limit object – the so called graphon.

In [STH18], the authors provide generalization bounds that are comparable to VC-dimension bounds known for CNNs. These bounds are improved in [GJJ20], which provides the first data dependent generalization bounds for MPNNs with sum aggregation that are comparable to Rademacher bounds for recurrent neural networks. [LUZ21] derive a generalization bound via a PAC-Bayesian approach that is governed by the maximum node degree and spectral norms of the weights. [VZ19a] consider generalization abilities of single-layer spectral GCNNs for node-classification task and provide a generalization bound that is directly proportional to the largest eigenvalue of the graph Laplacian. Another paper of this flavor is [YFM+21], showing that certain MPNNs (with sum aggregation) do not generalize from small to large graphs.

1.3 Main Contributions

We follow the route of [KBV20] and consider graphs as discretizations of continuous spaces in our analysis, called random graph models (RGM, see Definition 2.3). We introduce a continuous version of message passing neural networks – the realization of MPNNs on random graph models, which we call cMPNNs. Such cMPNNs are seen as limit objects of graph MPNNs, when the number of graph nodes goes to infinity. We prove, up to our knowledge, the first convergence result of the graph MPNN to the corresponding cMPNN as the number of nodes increases, which is uniform in the choice of the MPNN.

For the generalization analysis, we assume that the data distribution $\mu_{\mathcal{G}}$ represents graphs which are randomly sampled from a collection of template RGMs, with a random number of nodes. Using our convergence results, we can then prove that the generalization error between the training set and the true distribution is small. Here, we give the following informal version of Theorem 3.3.

Theorem 1.1 (Informal version of Theorem 3.3)**.** *Consider a graph classification task with m training samples $\mathcal{T} = (\mathbf{x}^i = (G^i, \mathbf{f}^i), \mathbf{y}^i)_{i=1}^m$ drawn i.i.d. from the data distribution $\mu_{\mathcal{G}}(\mathbf{x}, \mathbf{y})$ on a metric-measure space χ of dimension D_χ. Suppose that the size N of each graph in $\mathcal{T}$ is drawn from a distribution ν. Then*

$$\mathbb{E}_{\mathcal{T}\sim\mu_{\mathcal{G}}^m}\left[\sup_{\Theta}\left(R_{emp}(\Theta) - R_{exp}(\Theta)\right)^2\right] \leq \frac{C}{m}\mathbb{E}_{N\sim\nu}\left[N^{-\frac{1}{D_\chi+1}}\right].$$

The constant C represents the complexity of the hypothesis space of the network, via the Lipschitz constants of the message and update functions and the depth of the MPNNs.

Theorem 3.3 shows how we can use fewer graphs m than model complexity C when training MPNNs if the graphs are sufficiently large.

2 Preliminaries

A weighted graph $G = (V, \mathbf{W}, E)$ with N nodes is a tuple, where $V = \{1, \ldots, N\}$ is the node set. The edge set is given by $E \subset V \times V$, where $(i, j) \in E$ if node i and j are connected by an edge. $\mathbf{W} = (w_{k,l})_{k,l}$ is the weight matrix, assigning the weight $w_{i,j}$ to the edge $(i, j) \in E$, and assigning zero if (i, j) is not an edge. The degree d_i of a node i is defined as $\mathrm{d}_i = \sum_{j=1}^N w_{i,j}$. If G is a simple graph, i.e., a weighted graph with $\mathbf{W} \in \{0, 1\}^{N\times N}$, the degree d_i is the number of nodes connected to node i by an edge. We define a *graph signal* $\mathbf{f} : V \to \mathbb{R}^F$ as a function that maps nodes to their features in $\mathbb{R}^F$, where $F \in \mathbb{N}$ is the feature dimension. The signal $\mathbf{f}$ can be represented by a matrix $\mathbf{f} = (\mathbf{f}_1, \ldots, \mathbf{f}_N) \in \mathbb{R}^{N\times F}$, where $\mathbf{f}_i \in \mathbb{R}^F$ is the feature at node i. We also call $\mathbf{f}$ a *(graph) feature map*.

For a random variable Y distributed according to κ, and a function F of Y, we denote by $\mathbb{E}_{Y\sim\kappa}[F(Y)]$ the expected value of $F(Y)$. Similarly, we denote by $\mathrm{Var}_{Y\sim\kappa}[F(Y)]$ the variance of $F(Y)$.

2.1 Message Passing Graph Neural Networks

Message passing graph neural networks (gMPNNs) are defined by realizing an architecture of a *message passing neural network (MPNN)* on a graph. MPNNs are defined independently of a particular graph.

Definition 2.1. *Let $T \in \mathbb{N}$ denote the number of layers. For $t = 1, \ldots, T$, let $\Phi^{(t)} : \mathbb{R}^{2F_{t-1}} \to \mathbb{R}^{H_{t-1}}$ and $\Psi^{(t)} : \mathbb{R}^{F_{t-1}+H_{t-1}} \to \mathbb{R}^{F_t}$ be functions that we call the* message *and* update *functions, where $F_t \in \mathbb{N}$ is called the feature dimension of layer t. The corresponding* message passing neural network (MPNN) Θ *is defined to be the sequence*

$$\Theta = ((\Phi^{(t)})_{t=1}^T, (\Psi^{(t)})_{t=1}^T).$$

The message and the update function in Definition 2.1 are often defined as multi-layer-perceptrons (MLPs). In a MPNNs, messages are sent between nodes and aggregated. An *aggregation scheme* is a permutation invariant function that takes the collection of features in the edges of each node and computes a new nodes feature. In this paper, we consider MPNNs with *mean aggregation*. Then, a gMPNN processes graph signals by realizing a MPNN on the graph as follows.

Definition 2.2. *Let $G = (V, \mathbf{W})$ be a weighted graph and Θ be a MPNN, as defined in Definition 2.1. For each $t \in \{1, \ldots, T\}$, we define the* gMPNN $\Theta_G^{(t)}$ *as the mapping that maps input graph signals $\mathbf{f} = \mathbf{f}^{(0)} \in \mathbb{R}^{N\times F_0}$ to the features in the t-th layer by*

$$\Theta_G^{(t)} : \mathbb{R}^{N\times F_0} \to \mathbb{R}^{N\times F_t}, \quad \mathbf{f} \mapsto \mathbf{f}^{(t)} = (\mathbf{f}_i^{(t)})_{i=1}^N,$$

where $\mathbf{f}^{(t)} \in \mathbb{R}^{N\times F_t}$ are defined sequentially by

$$\mathbf{m}_i^{(t)} := \frac{1}{\mathrm{d}_i}\sum_{j=1}^N w_{i,j}\Phi^{(t)}(\mathbf{f}_i^{(t-1)}, \mathbf{f}_j^{(t-1)})$$
$$\mathbf{f}_i^{(t)} := \Psi^{(t)}(\mathbf{f}_i^{(t-1)}, \mathbf{m}_i^{(t)}),$$

for every $i \in V$. We call $\Theta_G := \Theta_G^{(T)}$ a message passing graph neural network (gMPNN).

Given a MPNN Θ as defined in Definition 2.1, the output $\Theta_G(\mathbf{f}) \in \mathbb{R}^{N \times F_T}$ is a graph signal. In graph classification or regression, the network should output a single feature for the whole graph. Hence, the output of a gMPNN after *global pooling* is a single vector $\Theta_G^P(\mathbf{f}) \in \mathbb{R}^{F_T}$, defined by

$$\Theta_G^P(\mathbf{f}) = \frac{1}{N} \sum_{i=1}^{N} \Theta_G(\mathbf{f})_i.$$

For brevity, in this paper we typically do not distinguish between a MPNN and its realization on a graph.

2.2 Random Graph Models

Let (χ, d, μ) be a metric-measure space, where χ is a set, d is a metric and μ is a probability Borel measure.

A *kernel* (also called a *graphon*), is a measurable mapping $W : \chi \times \chi \to \mathbb{R}$. The points $x \in \chi$ of the metric space are seen as the nodes of a continuous model, and the kernel is seen as a continuous version of a weight matrix. Kernels are treated as generative graph models using the following definition.

Definition 2.3. *A* random graph model (RGM) *on (χ, d, μ) is defined as a pair (W, f) of a kernel $W : \chi \times \chi \to \mathbb{R}$ and a measurable function $f : \chi \to \mathbb{R}$ called a* metric-space signal*. We define a* random graph *with corresponding node features $(G, \mathbf{f})$ by sampling N i.i.d. random points $X_1, \dots, X_N$ from χ, with probability density μ, as the nodes of G. The weight matrix $\mathbf{W} = (w_{i,j})_{i,j}$ of G is defined by $w_{i,j} = W(X_i, X_j)$ for $i, j = 1, \dots, N$. The graph signal $\mathbf{f}$ is defined by $\mathbf{f}_i = f(X_i)$. We say that $(G, \mathbf{f})$ is* drawn *from W, and denote $(G, \mathbf{f}) \sim (W, f)$.*

2.3 Continuous Message Passing Neural Networks

Given a MPNN, we define *continuous message passing neural networks (cMPNNs)* that act on kernels and metric-space signals $f : \chi \to \mathbb{R}^F$, by replacing the graph node features and the aggregation scheme in (2.2) by continuous counterparts. Let W be a kernel. We define the *kernel degree* of W at $x \in \chi$ by

$$\mathrm{d}_W(x) = \int_\chi W(x, y) d\mu(y). \tag{4}$$

Consider a message signal $U : \chi \times \chi \to \mathbb{R}^H$, where $U(x, y)$ is interpreted as a message sent from the point y to the point x in χ. We define the continuous mean aggregation of U by

$$M_W(U)(x) = \int_\chi \frac{W(x, y)}{\mathrm{d}_W(x)} U(x, y) d\mu(y).$$

Given the messages $U(x, y) = \Phi(f(x), f(y))$, where $\Phi : \mathbb{R}^{2F} \to \mathbb{R}^H$, we have

$$M_W(U)(x) = M_W\Big(\Phi\big(f(\cdot), f(\cdot\cdot)\big)\Big)(x) = \int_\chi \frac{W(x, y)}{\mathrm{d}_W(x)} \Phi\big(f(x), f(y)\big) d\mu(y).$$

By abuse of notation, we often denote in short $\Phi(f, f) := \Phi\big(f(\cdot), f(\cdot\cdot)\big)$.

By replacing mean aggregation by continuous mean aggregation in Definition 2.2, the same message and update functions that define a graph MPNN can also process metric-space signals.

Definition 2.4. *Let W be a kernel and Θ be a MPNN, as defined in Definition 2.1. For each $t \in \{1, \dots, T\}$, we define $\Theta_W^{(t)}$ as the mapping that maps the input signal to the signal in the t-th layer by*

$$\Theta_W^{(t)} : L^2(\chi) \to L^2(\chi), \quad f \mapsto f^{(t)}, \tag{5}$$

where $f^{(t)}$ are defined sequentially by

$$\begin{aligned} g^{(t)}(x) &= M_W\Big(\Phi^{(t)}\big(f^{(t-1)}, f^{(t-1)}\big)\Big)(x) \\ f^{(t)}(x) &= \Psi^{(t)}\Big(f^{(t-1)}(x), g^{(t)}(x)\Big) \end{aligned} \tag{6}$$

and $f^{(0)} = f : \chi \to \mathbb{R}^{F_0}$ is the input metric-space signal. We call $\Theta_W := \Theta_W^{(T)}$ a continuous message passing neural network (cMPNN).

As with graphs, the output of a cMPNN Θ_W on a metric-space signal $f : \chi \to \mathbb{R}^{F_0}$ is another metric-space signal $\Theta_W(f) : \chi \to \mathbb{R}^{F_T}$. The output of a cMPNN after *global pooling* is a single vector $\Theta_W^P(f) \in \mathbb{R}^{F_T}$, defined by $\Theta_W^P(\mathbf{f}) = \int_\chi \Theta_W(f)(x) d\mu(x)$.

2.4 Data Distribution for Graph Classification Tasks

In the following, we consider a training data $\mathcal{T} = \left(\mathbf{x}^i = (G^i, \mathbf{f}^i), \mathbf{y}^i\right)_{i=1}^m$ of graphs G^i, graph signals $\mathbf{f}^i$, and corresponding values $\mathbf{y}^i$ that can represent the classes of the graph-signal pairs. The training data is assumed to be drawn i.i.d. from a distribution $\mu_\mathcal{G}(\mathbf{x}, \mathbf{y})$ that we describe next.

In this paper, we focus on classification tasks. More precisely we have classes $j = 1, \ldots, \Gamma$, each represented by a RGM (W^j, f^j) on a metric-measure space (χ^j, d^j, μ^j). In fact, we suppose that each class corresponds to a set of metric spaces. For example, a graph representing a chair can be sampled from a template of either an office chair, a garden chair, a bar stool, etc., and each of these is represented by a metric space. For simplicity of the exposition, we however treat every template metric space as its own class. This does not affect our analysis.

The distribution $\mu_\mathcal{G}(\mathbf{x}, \mathbf{y})$ is defined via the following procedure of data sampling. For sampling one graph, first, choose a class with probability γ_j, i.e., for $(\mathbf{x}, \mathbf{y}) \sim \mu_\mathcal{G}$ and $j = 1, \ldots, \Gamma$, $\gamma_j = \mathbb{P}(\mathbf{y} = j)$. Independently of the choice of the class, choose the number of nodes $N \sim \nu$, where ν is a discrete distribution on $N \in \mathbb{N}$. After choosing a class $\mathbf{y} \in \{1, \ldots, \Gamma\}$ and the graph size N, a random graph $(G, \mathbf{f}) \sim (W^\mathbf{y}, f^\mathbf{y})$ with N nodes is drawn from the space $\chi^\mathbf{y}$ with probability density of the nodes $(\mu^\mathbf{y})^N$.

The notation $\mathcal{T} \sim \mu_\mathcal{G}^m$ describes a dataset $\mathcal{T}$ consisting of m samples $(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^m, \mathbf{y}^m)$ drawn i.i.d. from $\mu_\mathcal{G}$. We refer to Subsection C.1 in the appendix for a detailed definition of the distribution $\mu_\mathcal{G}$.

3 Convergence and Generalization of MPNNs

In this section, we provide our main results on convergence (Subsection 3.1) and generalization (Subsection 3.2) of MPNNs. For $z \in \mathbb{R}^F$, we define $\|z\|_\infty = \max_{j=1,\ldots,F} |z_j|$. Given a metric space $(\mathcal{Y}, d_\mathcal{Y})$, we define the infinity norm of a vector valued function $g : \mathcal{Y} \to \mathbb{R}^F$ by $\|g\|_\infty = \max_{j=1,\ldots,F} \operatorname{ess\,sup}_{y \in \mathcal{Y}} |(g(y))_j|$. The function g is called *Lipschitz continuous* if there exists a constant $L_g \in \mathbb{R}$ such that for all $y, y' \in \mathcal{Y}$,

$$\|g(y) - g(y')\|_\infty \leq L_g d_\mathcal{Y}(y, y').$$

If the domain $\mathcal{Y}$ is Euclidean, we always endow it with the L^∞-metric.

We measure the error between the output of a continuous MPNN and a gMPNN after pooling as follows. Given a graph signal $\mathbf{f} \in \mathbb{R}^{N \times F}$ and a metric-space signal $f : \chi \to \mathbb{R}^F$, both the graph and the continuous MPNN map to the same output space, i.e, $\Theta_W^P(f), \Theta_G^P(\mathbf{f}) \in \mathbb{R}^{F_T}$. Namely, the output dimension of Θ^P is independent of the random graph model it is realized on and also independent of the graph. Hence, we define the error to be the supremum norm $\|\Theta_W^P(f) - \Theta_G^P(\mathbf{f})\|_\infty$. We define the ε-covering numbers of the metric space χ, denoted by $\mathcal{C}(\chi, \varepsilon, d)$, as the minimal number of balls of radius ε required to cover χ.

For every $j = 1, \ldots, \Gamma$, we make the following assumptions, which hold for the remainder of the paper. We assume that there exist constants $C_{\chi^j}, D_{\chi^j} > 0$ such that

$$\mathcal{C}(\chi^j, \varepsilon, d) \;\leq\; C_{\chi^j}\, \varepsilon^{-D_{\chi^j}} \tag{7}$$

for every $\varepsilon > 0$. Denote $D_\chi = \max_j D_{\chi^j}$ and $C_\chi = \max_j C_{\chi^j}$ Such constants exist for every metric space with finite Minkowski dimension (see Appendix A). We assume that $\operatorname{diam}(\chi^j) := \sup_{x,y \in \chi^j} \{d(x, y)\} \leq 1$. Further, we only consider kernels W^j such that there exists a constant $\mathrm{d}_{\min} > 0$ satisfying

$$\mathrm{d}_{W^j}(x) \geq \mathrm{d}_{\min}, \tag{8}$$

where the kernel degree d_{W^j} is defined in (4). We moreover assume that $W^j(x, \cdot)$ and $W^j(\cdot, x)$ are Lipschitz continuous (with respect to its second and first variable, respectively) with Lipschitz constant L_{W^j} for every $x \in \chi$. We also assume that the metric-space signal $f^j : \chi \to \mathbb{R}^F$ is

Lipschitz continuous. Since the diameter of χ^j is finite, this means that $f^j \in L^\infty(\chi)$. We consider the following class of MPNNs

$$\mathrm{Lip}_{L,B} := \Big\{\Theta = \big((\Phi^{(l)})_{l=1}^T, (\Psi^{(l)})_{l=1}^T\big) \;\Big|\; \forall l = 1,\dots,T,\;\; \Phi^{(l)} : \mathbb{R}^{F_l} \to \mathbb{R}^{H_l} \text{ and } \Psi^{(l)} : \mathbb{R}^{F_l+H_l} \to \mathbb{R}^{F_{l+1}} \text{ satisfy } L_{\Phi^{(l)}}, L_{\Psi^{(l)}} \le L \text{ and } \|\Phi^{(l)}(0,0)\|_\infty, \|\Psi^{(l)}(0,0)\|_\infty \le B\Big\}.$$

3.1 Convergence

In this subsection we show that the error between the cMPNN and the according gMPNN decays when the number of nodes increases.

Theorem 3.1. *Let $W : \chi^2 \to \mathbb{R}$ be a Lipschitz continuous kernel with Lipschitz constant L_W, where the metric space χ satisfies (7) with respect to the constants $C_\chi, D_\chi > 0$, and W satisfies (8). Consider a graph $(G, \mathbf{f}) \sim (W, f)$ with N nodes $X_1, \dots, X_N$ drawn i.i.d. from χ with probability density μ. Then, for every Lipschitz continuous $f : \chi \to \mathbb{R}^F$,*

$$\mathbb{E}_{X_1,\dots,X_N\sim\mu^N}\left[\sup_{\Theta\in\mathrm{Lip}_{L,B}} \left\|\Theta_G^P(\mathbf{f}) - \Theta_W^P(f)\right\|_\infty^2\right] \le C'\big(1 + \|f\|_\infty^2 + L_f^2\big)\frac{\log(N)}{N^{1/(D_\chi+1)}} + \mathcal{O}(N^{-1}),$$

where C' is defined in Subsection B.2 of the appendix.

Remark 3.2. *The constant C' in Theorem 3.1 depends polynomially on the Lipschitz constants $L_{\Phi^{(l)}}$ and $L_{\Psi^{(l)}}$ of the message and update functions $\Phi^{(l)}$ and $\Psi^{(l)}$, on the so called* formal biases *$\|\Phi^{(l)}(0,0)\|_\infty$ and $\|\Psi^{(l)}(0,0)\|_\infty$, on $\|W\|_\infty$, on the Lipschitz constant L_W of W, on $\sqrt{\log(C_\chi)} + \sqrt{D_\chi}$, and on $\frac{1}{\mathrm{d}_{\min}}$, where the degree of the polynomial is T. A regularization of these constants can alleviate the exponential dependency of the bound on T.*

The proof of Theorem 3.1 is given in Subsection B.2 of the appendix.

Discussion and Comparison to other Convergence Results The work closest related to our convergence results is [KBV20], where the authors show convergence of a fixed spectral GCNN to its continuous counterpart with comparable regularity assumptions as in Theorem 3.1. Our result holds for MPNNs, which are more general than spectral GCNNs. Moreover, our bound is uniform in the choice of the MPNN Θ. This last property is essential for leveraging the convergence result to derive a generalization error. Indeed, using the bound from [KBV20], for each MPNN Θ there is a different high probability event $\mathcal{E}_\Theta$ where the convergence error is small. However, the trained MPNN $\Theta = \Theta_\mathcal{T}$ depends on the dataset $\mathcal{T}$ and cannot be fixed in the analysis. Hence, we would need to intersect all events $\bigcap_\Theta \mathcal{E}_\Theta$ to guarantee a small convergence error of the trained network $\Theta_\mathcal{T}$, which would not result in an event of high probability.

3.2 Generalization

In this subsection, we state the main result of our paper, which provides a non-asymptotic bound on the generalization error of MPNNs, as defined in (3). We consider a graph classification task with a training set $\mathcal{T} = (\mathbf{x}^i = (G^i, \mathbf{f}^i), \mathbf{y}^i)_{i=1}^m$ and Γ classes. The graphs and graph features in $\mathcal{T}$ are drawn i.i.d. from a probability distribution $\mu_\mathcal{G}(\mathbf{x}, \mathbf{y})$ as described in Subsection 2.4. We recall that the distribution that samples the size of the graph is denote by ν.

Given a MPNN with pooling, Θ^P, and its output dimension $\mathbb{R}^{F_T}$, we consider a non-negative loss function $\mathcal{L} : \mathbb{R}^{F_T} \times \{1, \dots, \Gamma\} \to [0, \infty)$. Additionally, we assume that $\mathcal{L}$ is Lipschitz continuous with Lipschitz constant $L_\mathcal{L}$. Note that although the cross-entropy loss, a popular choice for loss function in classification tasks, is not Lipschitz-continuous, cross-entropy composed on softmax is.

Theorem 3.3. *There exists a constant $C > 0$ such that*

$$\mathbb{E}_{\mathcal{T}\sim p^m}\left[\sup_{\Theta\in\mathrm{Lip}_{L,B}}\Big(R_{emp}(\Theta^P) - R_{exp}(\Theta^P)\Big)^2\right] \le \frac{2^\Gamma 8\|\mathcal{L}\|_\infty^2\pi}{m} + \frac{2^\Gamma L_\mathcal{L}^2 C}{m}\sum_j \gamma_j\big(1 + \|f^j\|_\infty^2 + L_{f^j}^2\big)\cdot\left(\mathbb{E}_{N\sim\nu}\left[\frac{1}{N} + \frac{1+\log(N)}{N^{1/(D_{\chi^j}+1)}} + \mathcal{O}\left(\exp(-N)N^{3T-\frac{3}{2}}\right)\right]\right),$$

Table 1: Comparison of generalization bounds for GNNs. We consider the following formula for a generic generalization bound: $GE \leq m^{-1/2}A(d,N)B(h)C(L,T) + Em^{-1/2}$, where m is the samples size, T is the depth, L is the bound of the Lipschitz constants of the message and update functions, h is the maximum hidden dimension, d is the average node degree and N is the graphs size and E is a term that does not depend on the model complexity.

	$A(d,N)$	$B(h)$	$C(L)$
VC-Dimension [STH18]	$\mathcal{O}(\log(N)N)$	$\mathcal{O}(h^4)$	-
Rademacher Complexity [GJJ20]	$\mathcal{O}(d^{T-1}\sqrt{\log(d^{2T-3})})$	$\mathcal{O}(h\sqrt{\log(h)})$	$\mathcal{O}(L^{2T})$
PAC-Bayesian [LUZ21]	$\mathcal{O}(d^{T-1})$	$\mathcal{O}(\sqrt{h\log(h)})$	$\mathcal{O}(L^{2T})$
Ours	$\mathcal{O}(\mathbb{E}_{N\sim\nu}[\log(N)N^{-\frac{1}{2(D_\chi+1)}}])$	$\mathcal{O}(1)$	$\mathcal{O}(L^{2T})$

where C is specified in Subsection C.2 of the appendix.

The proof of Theorem 3.3 is given in Subsection C.2 of the appendix.

Remark 3.4. *The constant C in Theorem 3.3 represents the complexity of the class $\mathrm{Lip}_{L,B}$ and can be bounded similarly to the constant C' from Theorem 3.1, as described in Remark 3.2. We summarize its dependencies on the parameters of the MPNN and the RGM by $\sqrt{C} \lesssim BL^{2T}\frac{1}{\mathrm{d}_{\min}^{T+1}}\max_{j=1,\dots,\Gamma}\left(\sqrt{\log(C_{\chi^j})}+\sqrt{D_{\chi^j}}\right)L_{W^j}\|W^j\|_\infty^T$ and refer to Subsection C.3 of the appendix for more details. Similarly to Remark 3.2 the exponential dependency of the constant C in Theorem 3.3 on the depth T and the polynomial dependency on the uniform Lipschitz bound L can be alleviated by regularizing the latter. We also note that the exponential dependency on the number of classes Γ in Theorem 3.3 can be eliminated by assuming that the data is representative, i.e., if the number of training samples that fall into class $j = 1,\dots,\Gamma$ is deterministically $\gamma_j m$.*

The term $\frac{2^\Gamma 8\|\mathcal{L}\|_\infty^2\pi}{m}$ in Theorem 3.3 does not depend on the model complexity and is typically much smaller than the second term. Hence, it does not affect bias–variance tradeoff considerations, and can be ignored in the situation where $m \gg C\mathbb{E}_{N\sim\nu}[\log(N)N^{-\frac{1}{D_\chi+1}}] \gg 1$. Theorem 3.3 allows us to think not just about graphs as samples, but also about individual nodes as samples. However, nodes are correlated with their neighbors, and the higher the dimension D_χ is, the larger the neighborhoods are. This is why the dependency on the number of nodes is $N^{-\frac{1}{2(D_\chi+1)}}$ and not $N^{-1/2}$. Still, this dependency of the bound on N explains one way in which we train on less graphs than model complexity and still generalize well. Another insight is that the generalization bound becomes smaller the smaller the Lipschitz constants of the message and update functions (see Remark 3.4). This indicates that regularization methods like weight decay promote generalization.

Comparison to other generalization bounds in graph classification We compare our generalization bound with other generalization bounds derived by bounding the VC-dimension [STH18], the Rademacher complexity [GJJ20], and using a PAC-Bayesian approach [LUZ21]. We do not compare with [VZ19b] since they derive generalization bounds for single-layered MPNNs in node-classification tasks. Hence, the role of depth is unexplored. Furthermore, their bound scales as $\mathcal{O}(\lambda_{\max}^{2T}/m)$, where T is the number of SGD steps and $\lambda_{\max}$ is the largest eigenvalue of the graph Laplacian. Hence, the generalization bound can increase monotonically for increasing T (see [LUZ21] for more details). We summarize the comparison in Table 1 and provide more details, specially on the comparability, in Subsection C.4 of the appendix.

Our analysis derives a generalization bound on MPNNs that has essentially the same dependency on the sample size m (up to a logarithmic factor), but does not directly depend on the number of hidden units. We emphasis that our bound depends on negative moments of the expected node size N. In contrast, the VC-dimension based bound [STH18] scales as $\mathcal{O}(\log(N)N)$, the Rademacher complexity based bound [GJJ20] scales as $\mathcal{O}(d^{T-1}\sqrt{\log(d^{2T-3})})$, and the PAC-Bayesian approach based bound [LUZ21] scales as $\mathcal{O}(d^{T-1})$, where d denotes the maximum node degree.

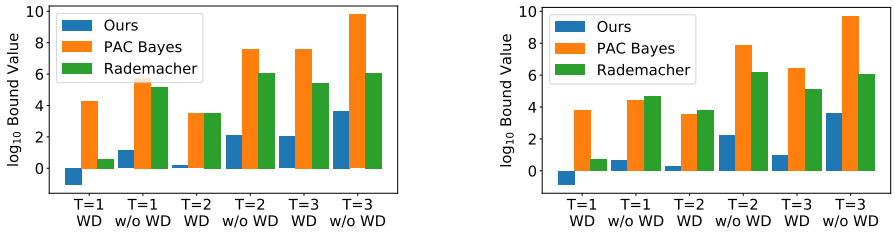

Figure 1: Generalization bounds given by our theory, PAC-Bayes [LUZ21] and Rademacher complexity [GJJ20] on a binary classification problem over Erdös-Rényi and SBM (left) and Erdös-Rényi and a geometric graph (right). Training is done with weight decay (WD) and without weight decay (w/o WD), and on three models with $T = 1, 2$ and 3 layers.

4 Numerical Experiments

We give empirical evaluations of our generalization bound in comparison to the PAC-Bayesian based bound [LUZ21] and the Rademacher complexity based bound [GJJ20]. We note that the VC dimension bound of [STH18] is written in O notations and hence cannot be quantitatively evaluated. We experiment on a synthetic dataset of 100K random graphs of 50 nodes, sampled from three different RGMs: the Erdös-Rényi model (ERM) with edge probability 0.4, a smooth version of a stochastic block model (SBM), based on the kernel $K(x, y) = \sin(2\pi x)\sin(2\pi y)/2\pi + 0.25$ on $[0, 1]^2$, and a geometric graph with kernel $K(x, y) = \exp(-|x - y|^2)$. The corresponding signals are given in Appendix D.2.1. Each RGM represents one class in three binary classification problems, comparing all pairs of RGMs. For the MPNN we consider GraphSAGE [HYL17] with mean aggregation, and number of layers $T = 1, 2$ or 3, implemented using Pytorch Geometric [FL19]. We consider a maximal hidden dimension of 128. In Appendix D we give more details and also consider synthetic data sampled from additional RGMs.

Our generalization bound becomes smaller the smaller the Lipschitz constants of the message and update functions are. To control the Lipschitz constants, we consider two learning settings. First, we train with weight decay regularization, which decreases the Lipschitz bounds, and second, we train with no regularization. For each setting (each choice of the number of layers and regularization) we train the MPNN, and read the resulting Lipschitz constants of the network. We then plug all constants into our generalization bound formula (see Theorem C.7 in the appendix for the full formula), and into the generalization bound formulas of the PAC-Bayes and Rademacher bounds (see Appendix C.4 for the formulas). The results are reported in Figure 1. We observe that our generalization bounds are orders of magnitude smaller than the other works. In fact, theoretical generalization bounds typically teach us about the asymptotic behavior of generalization, and about the hyperparameters that affect generalization, but rarely give realistic numerical bounds (less than 1) that guarantee generalization. Nevertheless, in one of the scenarios (one layer MPNN) our theory gives the bounds 0.08911 and 0.13299 (respectively in the two datasets of Figure 1), which guarantees generalization in practice.

5 Conclusion

In this paper we proved that MPNNs with mean aggregation generalize from training to test data in classification tasks, if the graphs are sampled from RGMs that represent the different classes. This follows from the fact that the MPNN on sampled graphs converges to the MPNN on the RGM when the number of nodes goes to infinity. Our generalization bounds become smaller the larger the graphs, which gives one explanation to how MPNNs with high complexity can generalize well from a relatively small dataset of large graphs. We observe two main limitations of our current model. First, the dependency of the generalization bound on the size of the graph N is $\mathcal{O}(N^{-\frac{1}{2(D_\chi+1)}})$, which is typically slower than the observed decay in experiments (See Appendix D.1). One potential future direction is to improve this dependency using a more sophisticated models of the trained network and of the message and update functions. Secondly, our model of the data is somewhat limited. One future direction is to allow deformations of the RGMs, to consider a continuum of RGMs instead of a finite set, and to consider sparse graphs.

References

[BBL+17] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, Jul 2017.

[DDS16] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2702–2711. JMLR.org, 2016.

[FL19] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[FML+19] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference*, WWW '19, page 417–426, New York, NY, USA, 2019. Association for Computing Machinery.

[GBR20] Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Stability properties of graph neural networks. *IEEE Transactions on Signal Processing*, 68:5680–5695, 2020.

[GJJ20] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3419–3430. PMLR, 13–18 Jul 2020.

[HYL17] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[KBV20] Nicolas Keriven, Alberto Bietti, and Samuel Vaiter. Convergence and stability of graph convolutional networks on large random graphs. *stat*, 1050:23, 2020.

[KTD21] Henry Kenlay, Dorina Thanou, and Xiaowen Dong. Interpretable stability bounds for spectral graph filters. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021.

[KW16] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[LHB+21] Ron Levie, Wei Huang, Lorenzo Bucci, Michael Bronstein, and Gitta Kutyniok. Transferability of spectral graph convolutional neural networks. *Journal of Machine Learning Research*, 22(272):1–59, 2021.

[LIK19] Ron Levie, Elvin Isufi, and Gitta Kutyniok. On the transferability of spectral graph filters. In *13th International conference on Sampling Theory and Applications (SampTA)*. IEEE, 2019.

[Lov67] L. Lovász. Operations with structures. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(3):321–328, 1967.

[Lov12] László Lovász. *Large networks and graph limits*. Colloquium Publications, Budapest, 2012.

[LUZ21] Renjie Liao, Raquel Urtasun, and Richard Zemel. A {pac}-bayesian approach to generalization bounds for graph neural networks. In *International Conference on Learning Representations*, 2021.

[MFE+19] Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein. Fake news detection on social media using geometric deep learning. *arXiv preprint arXiv:1902.06673*, 2019.

[MLK21] Sohir Maskey, Ron Levie, and Gitta Kutyniok. Transferability of graph neural networks: an extended graphon approach. *arXiv preprint arXiv:2109.10096*, 2021.

[MRF+19] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4602–4609, Jul. 2019.

[Pen03] Mathew Penrose. *Random Geometric Graphs*. Oxford Scholarship Online, 2003.

[RGR21] Luana Ruiz, Fernando Gama, and Alejandro Ribeiro. Graph neural networks: Architectures, stability, and transferability. *Proceedings of the IEEE*, 109(5):660–682, 2021.

[RWR21] Luana Ruiz, Zhiyang Wang, and Alejandro Ribeiro. Graphon and graph neural network stability. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.

[STH18] Franco Scarselli, Ah Chung Tsoi, and Markus Hagenbuchner. The vapnik–chervonenkis dimension of graph and recursive neural networks. *Neural Networks*, 108:248–259, 2018.

[SYS+20] Jonathan M. Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M. Donghia, Craig R. MacNair, Shawn French, Lindsey A. Carfrae, Zohar Bloom-Ackermann, Victoria M. Tran, Anush Chiappino-Pepe, Ahmed H. Badran, Ian W. Andrews, Emma J. Chory, George M. Church, Eric D. Brown, Tommi S. Jaakkola, Regina Barzilay, and James J. Collins. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702.e13, 2020.

[Ver18] Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018.

[VW96] Aad W. Vaart and Jon A. Wellner. *Weak Convergence and Empirical Processes*. Springer New York, NY, 1996.

[VZ19a] Saurabh Verma and Zhi-Li Zhang. Stability and generalization of graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1539–1548, 2019.

[VZ19b] Saurabh Verma and Zhi-Li Zhang. Stability and generalization of graph convolutional neural networks, 2019.

[WHZ+18] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 839–848, New York, NY, USA, 2018. Association for Computing Machinery.

[WZL+18] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018.

[XHLJ19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[YFM+21] Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11975–11986. PMLR, 18–24 Jul 2021.

[YHC+18] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 974–983, New York, NY, USA, 2018. Association for Computing Machinery.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the conclusion in Subsection 5.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This is theoretical work with no foreseeable societal consequences
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Yes
   (b) Did you include complete proofs of all theoretical results? [Yes] The appendix includes complete proofs.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A] The experiments are very simple which is why we decided to not include the code.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A] We did not perform training
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report the standard error with respect to the randomness of our experiments.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A] We did not perform training, i.e., the experiments were toy examples and not computation heavy.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We refer to [FL19] for using Pytorch Geometric. We do not use any existing code, data or model.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We do not use real data. Only toy experiments on synthetic data.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We do not use real data. Only toy experiments on synthetic data.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We did not crowdsource or conduct research with human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]