Generalization of Graph Neural Networks is Robust to Model Mismatch

Zhiyang Wang*, Juan Cerviño†, Alejandro Ribeiro*

*University of Pennsylvania, USA   † Massachusetts Institute of Technology, USA
zhiyangw@seas.upenn.edu, jcervino@mit.edu, aribeiro@seas.upenn.edu

Abstract

Graph neural networks (GNNs) have demonstrated their effectiveness in various tasks supported by their generalization capabilities. However, the current analysis of GNN generalization relies on the assumption that training and testing data are independent and identically distributed (i.i.d). This imposes limitations on the cases where a model mismatch exists when generating testing data. In this paper, we examine GNNs that operate on geometric graphs generated from manifold models, explicitly focusing on scenarios where there is a mismatch between manifold models generating training and testing data. Our analysis reveals the robustness of the GNN generalization in the presence of such model mismatch. This indicates that GNNs trained on graphs generated from a manifold can still generalize well to unseen nodes and graphs generated from a mismatched manifold. We attribute this mismatch to both node feature perturbations and edge perturbations within the generated graph. Our findings indicate that the generalization gap decreases as the number of nodes grows in the training graph while increasing with larger manifold dimension as well as larger mismatch. Importantly, we observe a trade-off between the generalization of GNNs and the capability to discriminate high-frequency components when facing a model mismatch. The most important practical consequence of this analysis is to shed light on the filter design of generalizable GNNs robust to model mismatch. We verify our theoretical findings with experiments on multiple real-world datasets.

Introduction

Graph Neural Networks (GNNs) [16, 25, 45], as a deep learning model on graphs, have been unarguably one of the most recognizable architectures when processing graph-structured data. GNNs have achieved notable performances in numerous applications, such as recommendation systems [59], protein structure predictions [64], and multi-agent robotic control [18]. These outstanding results of GNNs depend on their empirical performances when *predicting* over unseen testing data. This is evaluated in theory with *statistical generalization analysis*, which quantifies the difference between the *empirical risk* (i.e. training error) and the *statistical risk* (i.e. testing error) in deep learning theory [22]. Recent works have focused on proving the generalization bounds of GNNs without any dependence on the underlying model responsible for generating the graph data [17, 46, 51]. Generalization analysis on graph classification is studied in a series of works when graphs are drawn from random limit models [28, 35, 37, 44]. In [53], the authors study the generalization of GNNs over graphs generated from an underlying manifold on both node and graph levels. These works assume that the training and testing graphs are generated from the same underlying model. In practice, there are inevitable scenarios with generative model mismatch between testing and training graphs [30]. Hence, it is of crucial to demonstrate the generalization ability of GNNs remains robust to generative model mismatch. This would provide a promising assurance that GNNs can maintain outstanding generalizable performance even in noisy environments.

The model mismatch may stem from disturbances during the graph generation process from the manifold. Moreover, the underlying manifold model is prone to undergo alternations and fluctuations in practical situations. While we take into account the underlying model mismatches, they can be interpreted as perturbations within the generated graph domain. Figure 1a displays the original manifold and a graph derived from it. Figure 1b illustrates a mismatched manifold, which leads to edge perturbations in the generated graph. Figure 1c shows the interpretation of perturbed manifold function values, resulting in node feature perturbations in the generated graph.
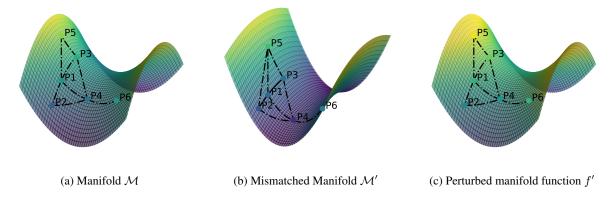
(a) Manifold $\mathcal{M}$ (b) Mismatched Manifold $\mathcal{M}'$ (c) Perturbed manifold function f'

Figure 1: Example of model mismatch. (a) The original manifold with a generated graph based on sampled points $(P1, \cdots, P6)$. (b) The mismatched manifold with the sampled points also shifted, resulting in a perturbed graph. (c) The manifold mismatch can be seen as the perturbation of manifold function values, which leads to perturbed node features on the generated graph.

We prove that the generalization of GNNs is robust to model mismatches based on the convergence of GNNs on generated graphs to neural networks on the manifold model [57], combined with the stability of the Manifold Neural Networks (MNNs) under manifold deformations [58]. Implementing low-pass and integral Lipschitz continuous filters (Definition 2) allows us to bound the generalization gap under model mismatches. This bound decreases with the number of nodes in the training graph and increases with both the mismatch size and the manifold dimension. The key insight from the bound is that a more robust generalization necessitates the cost of failing to discriminate high spectral components in the graph data.

Our main contributions are as follows:

1. We analyze the generalization of the GNNs when there exists a manifold model mismatch between the training data and testing data.

2. We determine the manifold mismatch as perturbations on the generated graphs, both as node feature perturbations and edge perturbations.

3. We propose that implementing continuity restrictions on the filters composing the GNN ensures the robust generalization in both node and graph classification tasks.

4. We observe a trade-off between robust generalization and discriminability through the generalization bound.

We conduct experiments on real-world datasets to validate our theoretical findings.

Related Works

In-Distribution Generalization of GNNs

Existing works on the in-distribution generalization of GNNs fall in node and graph level tasks. For node classification tasks of GNNs, there are works providing a generalization bound of GNNs based on a Vapnik-Chervonenkis dimension [46], algorithmic stability analysis [51, 65], PAC-Bayesian analysis [34] and Rademacher complexity [12]. For graph classification tasks of GNNs, the authors prove the generalization bound via Rademacher complexity [17] and PAC-Bayes analysis [21, 31]. The authors consider a continuous graph limit model to analyze the generalization of GNNs on graph classification in [28, 35, 37]. In [54], the authors prove the generalization of GNNs on graphs sampled from a manifold both for node and graph classification tasks. These works are considered only in the in-distribution case where the training and testing data are sampled from the same distribution.

Out-of-Distribution Generalization of GNNs

Several works have extensively addressed the out-of-distribution generalization of GNNs with graph enhancement methods. The authors in [49] propose a domain generalization framework for node-level tasks on graphs to address distribution shifts in node attribute distribution and graphic topology. In [13], the authors study the out-of-distribution generalization of GNNs on graph-level tasks with a causal representation learning framework. In [30] the authors handle graph distribution shifts in complex and heterogeneous situations of GNNs with a nonlinear graph representation decorrelation method. The authors in [63] propose a size generalization analysis of GNNs correlated to the discrepancy between local distributions of graphs. In our novel approach, we conceptualize the generative model mismatch as a distribution shift. Our findings can be further generalized into a theoretical framework for GNNs in manifold domain shift scenarios. In such cases, the generalization gap is directly proportional to the distance between the manifold models. This extension is discussed in more detail within the supplementary material.

GNNs and Manifold Neural Networks

Geometric deep learning has been proposed in [4] with neural network architectures on manifolds. The authors in [39] and [7] provide neural network architectures for manifold-valued data. In [58] and [56], the authors define convolutional operation over manifolds and see the manifold convolution as a generalization of graph convolution, which establishes the limit of neural networks on large-scale graphs as manifold neural networks (MNNs). The authors in [57] further establish the relationship between GNNs and MNNs with non-asymptotic convergence results for different graph constructions.

Neural Networks on Manifolds

Suppose there is a d-dimensional embedded Riemannian manifold $\mathcal{M} \subset \mathbb{R}^M$ which is compact, smooth and differentiable. A measure μ over $\mathcal{M}$ with density function $\rho : \mathcal{M} \to (0, \infty)$, which is assumed to be bounded as $0 < \rho_{min} \leq \rho(x) \leq \rho_{max} < \infty$ for each $x \in \mathcal{M}$. Data supported over the manifold is defined as a scalar function $f : \mathcal{M} \to \mathbb{R}$ [58] mapping datum value $f(x)$ to each point $x \in \mathcal{M}$. The manifold with density ρ is endowed with a weighted Laplace operator [19], which generalizes the Laplace-Beltrami operator as

$$\mathcal{L}f = -\frac{1}{2\rho}\text{div}(\rho^2 \nabla f), \tag{1}$$

where div denotes the divergence operator of $\mathcal{M}$ and ∇ denotes the gradient operator of $\mathcal{M}$ [4].

We consider square-integrable functions over $\mathcal{M}$, denoted as use $L^2(\mathcal{M})$. The inner product of functions $f, g \in L^2(\mathcal{M})$ is defined as

$$\langle f, g \rangle_{\mathcal{M}} = \int_{\mathcal{M}} f(x)g(x)\text{d}\mu(x), \tag{2}$$

while the L^2 norm is defined as $\|f\|_{\mathcal{M}}^2 = \langle f, f \rangle_{\mathcal{M}}$.

Manifold convolution is defined by aggregating the heat diffusion process over $\mathcal{M}$ with operator $\mathcal{L}$ [56, 58]. With the input manifold function $f \in L^2(\mathcal{M})$, the manifold convolution output is

$$g(x) = \mathbf{h}(\mathcal{L})f(x) = \sum_{k=0}^{K-1} h_k e^{-k\mathcal{L}}f(x). \tag{3}$$

Considering the frequency representation, the Laplace operator $\mathcal{L}$ has real, positive and discrete eigenvalues $\{\lambda_i\}_{i=1}^{\infty}$ as the Laplace operator is self-adjoint and positive-semidefinite. The eigenfunction associated with each eigenvalue is denoted as ϕ_i, i.e. $\mathcal{L}\phi_i = \lambda_i \phi_i$. The eigenvalues are ordered as $0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \ldots$, and the eigenfunctions are orthonormal and form the eigenbasis of $L^2(\mathcal{M})$. When mapping a manifold function onto one of the eigenbasis ϕ_i, we have the spectral component as $[\hat{f}]_i = \langle f, \phi_i \rangle_{\mathcal{M}}$. The manifold convolution can be written in the spectral domain point-wisely as

$$[\hat{g}]_i = \sum_{k=0}^{K-1} h_k e^{-k\lambda_i}[\hat{f}]_i. \tag{4}$$

Hence, the frequency response of manifold filter is given by $\hat{h}(\lambda) = \sum_{k=0}^{K-1} h_k e^{-k\lambda}$, depending only on the filter coefficients h_k and eigenvalues of $\mathcal{L}$ when $\lambda = \lambda_i$.

Manifold neural networks (MNNs) are built by cascading layers consisting of a bank of manifold filters and a pointwise nonlinearity function $\sigma : \mathbb{R} \to \mathbb{R}$, with the output of each layer written as

$$f_l(x) = \sigma\Big(\mathbf{h}_l(\mathcal{L})f_{l-1}(x)\Big). \tag{5}$$

To represent the MNN succinctly, we group all learnable parameters, and denote the mapping based on input manifold function $f \in L^2(\mathcal{M})$ to predict target function $g \in L^2(\mathcal{M})$ as $\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)$, where $\mathbf{H} \in \mathcal{H} \subset \mathbb{R}^P$ is a filter parameter set of the manifold filters. A positive loss function is denoted as $\ell(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f), g)$ to measure the estimation performance.

Geometric Graph Neural Networks

Suppose we can access a discrete set of sampled points over manifold $\mathcal{M}$. A graph $\mathbf{G}$ is generated based on a set of N i.i.d. randomly sampled points $X_N = \{x_N^1, x_N^2, \cdots, x_N^N\}$ according to measure μ over $\mathcal{M}$. Seeing these N sampled points as nodes, edges connect every pair of nodes (x_N^i, x_N^j) is connected with weight value $[\mathbf{W}_N]_{ij}, \mathbf{W}_N \in \mathbb{R}^{N \times N}$ determined by a function of their Euclidean distance $\|x_N^i - x_N^j\|$ [5], explicitly written as

$$[\mathbf{W}_N]_{ij} = \frac{\alpha_d}{(d+2)N\epsilon^{d+2}} \mathbb{1}_{[0,1]} \left(\frac{\|x_N^i - x_N^j\|}{\epsilon} \right), \tag{6}$$

where α_d is the volume of the d-dimensional Euclidean unit ball and $\mathbb{1}$ represents an indicator function. Based on this, the graph Laplacian can be calculated as $\mathbf{L}_N = \text{diag}(\mathbf{W}_N \mathbf{1}) - \mathbf{W}_N$. On this generated graph, graph data values are sampled from the functions over manifold $\mathcal{M}$ [8, 56]. Consider the input and target functions $f, g \in L^2(\mathcal{M})$, the sampled input and target functions $\mathbf{x}, \mathbf{y} \in L^2(X_N)$ over graph $\mathbf{G}$ can be written as

$$[\mathbf{x}]_i = f(x_N^i), \quad [\mathbf{y}]_i = g(x_N^i). \tag{7}$$

A convolutional filter on graph $\mathbf{G}$ can be extended from manifold convolution defined in (3) by replacing the Laplace operator with the graph Laplacian as

$$\mathbf{y} = \mathbf{h}(\mathbf{L})\mathbf{x} = \sum_{k=0}^{K-1} h_k e^{-k\mathbf{L}}\mathbf{x}. \tag{8}$$

We observe that this formation is accordant with the definition of the graph convolution [16] with graph shift operator as $e^{-\mathbf{L}}$. Replace $\mathbf{L}$ with eigendecomposition $\mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^H$, where $\mathbf{V}$ is the eigenvector matrix and $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues of $\mathbf{L}$ as the entries. The spectral representation of this filter on $\mathbf{G}$ is

$$\mathbf{V}^H \mathbf{h}(\mathbf{L}_\tau)\mathbf{x} = \sum_{k=1}^{K-1} h_k e^{-\mathbf{\Lambda}k}\mathbf{V}^H\mathbf{x} = \hat{h}(\mathbf{\Lambda})\mathbf{V}^H\mathbf{x}. \tag{9}$$

This analogously leads to a frequency response of this graph convolution, i.e. $\hat{h}(\lambda) = \sum_{k=0}^{K-1} h_k \lambda^k$, relating the input and output spectral components point-wisely.

Neural networks on $\mathbf{G}$ is composed of layers consisting of graph filters and point-wise nonlinearity σ, written as

$$\mathbf{x}_l = \sigma\Big(\mathbf{h}_l(\mathbf{L})\mathbf{x}_{l-1}\Big). \tag{10}$$

The mapping from input graph data $\mathbf{x} \in L^2(X_N)$ to predict $\mathbf{y} \in L^2(X_N)$ is denoted as $\mathbf{\Phi}(\mathbf{H}, \mathbf{L}, \mathbf{x})$ with $\mathbf{H} \in \mathcal{H} \subset \mathbb{R}^P$ which is trained to minimize the loss $\ell(\mathbf{\Phi}(\mathbf{H}, \mathbf{L}, \mathbf{x}), \mathbf{y})$.

Generalization of GNNs to model mismatch

Manifold Model Mismatch

A mismatched model of manifold $\mathcal{M}$ is denoted as $\mathcal{M}^\tau$ where τ maps each point $x \in \mathcal{M}$ to a displaced $\tau(x) \in \mathcal{M}^\tau$. We restrict this mismatch function class τ as it preserves the properties of $\mathcal{M}$ and denote the

curvature distance between x and the displaced $\tau(x)$ as dist$(x, \tau(x))$. The mismatch τ induces a tangent map $\tau_{*,x} : T_x\mathcal{M} \to T_{\tau(x)}\mathcal{M}$ which is a linear map between the tangent spaces [50]. With the coordinate description over $\mathcal{M}$, the tangent map $\tau_{*,x}$ can be exactly represented by the Jacobian matrix $J_x(\tau)$.

The manifold model mismatch can be seen as deformations to input manifold functions as shown in Figure 1c.

$$\mathcal{L}f(\tau(x)) = \mathcal{L}f'(x), \qquad x \in \mathcal{M}. \tag{11}$$

While the model mismatch can also be understood as deformations to the Laplacian operator as shown in Figure 1b.

$$\mathcal{L}f(\tau(x)) = \mathcal{L}_\tau f(x), \qquad x \in \mathcal{M}. \tag{12}$$

Generalization of GNNs on node-level

The generalization analysis is restricted to a finite-dimensional subset of $L^2(\mathcal{M})$, i.e. the functions over the manifold are bandlimited as defined in Definition 1.

Definition 1. *Function $f \in L^2(\mathcal{M})$ is bandlimited if there exists some $\lambda > 0$ such that for all eigenpairs $\{\lambda_i, \phi_i\}_{i=1}^\infty$ of the weighted Laplacian $\mathcal{L}$ when $\lambda_i > \lambda$, we have $\langle f, \phi_i \rangle_\mathcal{M} = 0$.*

Assumption 1. *(Lipschitz target function) The target function g is Lipschitz continuous, i.e., $|g(x) - g(y)| \leq C_g dist(x, y)$, for all $x, y \in \mathcal{M}$.*

We further assume that the filters in MNN $\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, \cdot)$ and GNN $\mathbf{\Phi}(\mathbf{H}, \mathbf{L}, \cdot)$ are low-pass and integral Lipschitz filters as defined in Definition 2. We note that this is a mild assumption as high frequency components on the graph/manifold implies that there exist functions with large variations in adjacent entries. This naturally generates instabilities that are more difficult to learn.

Definition 2. *A filter is a low-pass and integral Lipschitz filter if its frequency response satisfies*

$$\left| \hat{h}(\lambda) \right| = \mathcal{O}\left(\lambda^{-d} \right), \quad \lambda \to \infty, \tag{13}$$

$$\left| \hat{h}'(\lambda) \right| \leq C_L \lambda^{-d-1}, \quad \lambda \in (0, \infty). \tag{14}$$

with d denoted as the dimension of manifold $\mathcal{M}$ and C_L a positive continuity constant.

We note that with a smaller C_L, the filter function tends to be smoother especially in the high-spectrum domain. The filters fail to discriminate different high spectral components with similar frequency responses given to them.

In the neural network architectures, we consider assumptions on both the nonlinear activation function and the loss function as presented in Assumption 2 and 3 respectively. We note that these are reasonable assumptions as most activations (e.g. ReLU, modulus, sigmoid) and loss functions (e.g. L1 regression, Huber loss, quantile loss).

Assumption 2. *(Normalized Lipschitz activation functions) The activation function σ is normalized Lipschitz continuous, i.e., $|\sigma(a) - \sigma(b)| \leq |a - b|$, with $\sigma(0) = 0$.*

Assumption 3. *(Normalized Lipschitz loss function) The loss function ℓ is normalized Lipschitz continuous, i.e., $|\ell(y_i, y) - \ell(y_j, y)| \leq |y_i - y_j|$, with $\ell(y, y) = 0$.*

The generalization gap is evaluated between the *empirical risk* over the discrete graph model and the *statistical risk* over the manifold model, which has been studied in both node-level and graph level in previous works when no model mismatch is considered [54, 55].

Suppose the training graph $\mathbf{G}$ is generated from a manifold $\mathcal{M}$. The empirical risk that we train the GNN to minimize is therefore defined as

$$R_\mathbf{G}(\mathbf{H}) = \frac{1}{N} \sum_{i=1}^N \ell\left([\mathbf{\Phi}(\mathbf{H}, \mathbf{L}, \mathbf{x})]_i, [\mathbf{y}]_i \right). \tag{15}$$

The statistical risk over mismatched manifold $\mathcal{M}_\tau$ is

$$R_{\mathcal{M}^\tau}(\mathbf{H}) = \int_{\mathcal{M}^\tau} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}_\tau, f)(x), g(x) \right) d\mu_\tau(x). \tag{16}$$

The generalization gap under model mismatch is

$$GA_\tau = \sup_{\mathbf{H} \in \mathcal{H}} |R_{\mathcal{M}^\tau}(\mathbf{H}) - R_\mathbf{G}(\mathbf{H})|. \tag{17}$$

Theorem 1. *Suppose a neural network is equipped with filters defined in Definition 2 and normalized nonlinearites (Assumption 2). The neural network is operated on a graph $\mathbf{G}$ generated according to (6) from $\mathcal{M}$ and a mismatched manifold $\mathcal{M}^\tau$ with a bandlimited (Definition 1) input manifold function. Suppose the mismatch $\tau : \mathcal{M} \to \mathcal{M}$ satisfies $dist(x, \tau(x)) \leq \gamma$ and $\|J_x(\tau) - I\|_F \leq \gamma$ for all $x \in \mathcal{M}$. The generalization of neural network trained on $\mathbf{G}$ to minimize a normalized Lipschitz loss function (Assumption 3) holds in probability at least $1 - \delta$ that*

$$GA_\tau \leq C_1 \frac{\epsilon}{\sqrt{N}} + C_2 \frac{\sqrt{\log(1/\delta)}}{N} + C_3 \left(\frac{\log N}{N} \right)^{\frac{1}{d}} + C_4 \gamma, \tag{18}$$

with $\epsilon \sim \left(\frac{\log(C/\delta)}{N} \right)^{\frac{1}{d+4}}$, C_1 scaling with C_L, C_4 scaling with C_L and C_g. C_1 and C_3 depend on the geometry of $\mathcal{M}$.

Proof. See supplementary material for proof and the definitions of C_1, C_2, C_3 and C_4. □

Remark 1

This conclusion is ready to extend to multi-layer and multi-feature neural network architectures, as the neural network is cascaded by layers of filters and nonlinearities. The generalization error propagates across layers, leading to an increase in the generalization gap of multi-layer and multi-feature GNNs with the size of the architecture, which we will further verify in simulations.

Theorem 1 indicates that the generalization of GNNs is robust to model mismatches, with the generalization gap decreasing with the number of nodes N in the training graph. As more points are sampled from the manifold, the generated graph can better approximate the underlying manifold. This leads to a better generalization of the GNN to predict the unseen points over the manifold. The upper bound also increases with the dimension of the underlying manifold d, as higher dimension indicates higher model complexity. Specifically, the generalization gap increases with the mismatch size γ as the testing manifold is shifted more from the original manifold.
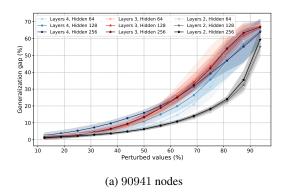
Remark 2

It is important to note that there is a trade-off involved in designing GNNs: achieving better generalization often comes at the cost of reduced discriminability. Specifically, using a smaller C_L leads to improved generalization and robustness by reducing generalization error. While this leads to smoother filter functions, which limits the GNN ability to discriminate between different spectral components. This reduced discriminability can negatively impact prediction performance. In essence, we can enhance better and more robust generalization capabilities of GNNs, but this improvement necessitates a compromise in their discriminative power.

Extension to Graph Classification

The generalization ability of GNN can be extended from the node level to the graph level, which indicates the manifold classification with approximated graph classification.

Suppose we have manifolds $\{\mathcal{M}_k^{\tau_k}\}_{k=1}^K$ with dimension d_k under a mismatch τ_k. Manifold $\mathcal{M}_k^{\tau_k}$ is labeled with $y_k \in \mathbb{R}$. The manifolds are smooth, compact, differentiable, and embedded in $\mathbb{R}^M$ with measure $\mu_{\tau_k, k}$. Each manifold $\mathcal{M}_k^{\tau_k}$ is equipped with a weighted Laplace operator $\mathcal{L}_{\tau_k, k}$. Assume that we can access N_k randomly sampled points according to μ_k over each manifold $\mathcal{M}_k$. Graphs generated based on these sampled points are denoted as $\{\mathbf{G}_k\}_{k=1}^K$ with graph Laplacians $\{\mathbf{L}_k\}_{k=1}^K$. A GNN $\mathbf{\Phi}(\mathbf{H}, \mathbf{L}_\cdot, \mathbf{x}_\cdot)$ is trained on these graphs with $\mathbf{x}_k$ denoted as the input data on graphs sampled from the data on manifolds $f_k \in L^2(\mathcal{M}_k)$. The output of the GNN is set as the average of the output values over all the nodes while the output of MNN $\mathbf{\Phi}(\mathbf{H}, \mathcal{L}_{\tau, \cdot}, f_\cdot)$ is the averaged value over the mismatched manifold. Loss function ℓ evaluates the performance of GNN and MNN by comparing the difference between the output label and the target label. The empirical risk that the GNN is trained to minimize is defined as

$$R_\mathbf{G}(\mathbf{H}) = \sum_{k=1}^K \ell \left(\frac{1}{N_k} \sum_{i=1}^{N_k} [\mathbf{\Phi}(\mathbf{H}, \mathbf{L}_k, \mathbf{x}_k)]_i, y_k \right). \tag{19}$$
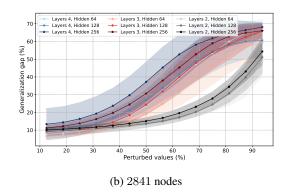
(a) 90941 nodes
(b) 2841 nodes

Figure 2: Generalization gap as a function of the percentage of perturbed feature values in node feature perturbation.

The statistical risk is defined based on the statistical MNN output and the target label over mismatched models as

$$R_{\mathcal{M}^\tau}(\mathbf{H}) = \sum_{k=1}^{K} \ell \left(\int_{\mathcal{M}_k} \mathbf{\Phi}(\mathbf{H}, \mathcal{L}_{\tau_k,k}, f_k)(x) \mathrm{d}\mu_{\tau_k}(x), y_k \right). \tag{20}$$

The generalization gap is therefore

$$GA_\tau = \sup_{\mathbf{H} \in \mathcal{H}} |R_{\mathcal{M}^\tau}(\mathbf{H}) - R_{\mathbf{G}}(\mathbf{H})|. \tag{21}$$

Theorem 2. *Suppose the GNN and MNN with filters defined in Definition 2 and the input manifold functions are bandlimited (Definition 1). Suppose the mismatches $\tau_k : \mathcal{M}_k \to \mathcal{M}_k$ where $\mathrm{dist}(x, \tau_k(x)) \leq \gamma$ and $\|J_x(\tau_k) - I\|_F \leq \gamma$ for all $x \in \mathcal{M}_k$ for $k = 1, 2, \cdots K$. Under Assumptions 2 and 3 it holds in probability at least $1 - \delta$ that*

$$GA_\tau \leq \sum_{k=1}^{K} \left(\frac{C_1}{\sqrt{N_k}} \epsilon_k + C_2 \frac{\sqrt{\log(1/\delta)}}{N_k} \right) + C_3 \sum_{k=1}^{K} \left(\frac{\log N_k}{N_k} \right)^{\frac{1}{d_k}} + KC_4\gamma,$$

with $\epsilon_k \sim \left(\frac{\log(C/\delta)}{N_k} \right)^{\frac{1}{d_k+4}}$, C_1 and C_4 scaling with C_L. C_1 and C_3 depend on the geometry of $\mathcal{M}$.

Proof. See supplementary material for proof and the definitions of C_1, C_2, C_3 and C_4. ☐

Theorem 2 indicates that a graph with large enough points sampled from each underlying manifold can approximately predict the label for mismatched manifolds. This shows that GNN trained over these generated graphs can generalize to classify unseen graphs generated from mismatched manifold models. The generalization gap on the graph level also decreases with the number of sampled points over each manifold. The generalization gap increases with the dimensions of the manifolds and the size of deformations. A similar trade-off phenomenon can also be observed.

Experiments

Node classification with Arxiv Dataset

In this section, we showcase the generalization properties of a trained GNN on a real-world dataset, OGBN-Arxiv [52]. The graph has $169,343$ nodes and $1,166,243$ edges, representing the citation network between computer science arXiv papers. The node features are 128 dimensional embeddings of the title and abstract of each paper [38]. The objective is to predict which of the 40 categories the paper belongs to. We consider two types of perturbations to model the impact of the underlying manifold model mismatch – node and edge perturbations.

In all cases, we train the GNN on the original graph, and we evaluate it on the perturbed graph. The generalization gap is measured by the difference between the training accuracy and the perturbed testing accuracy. For
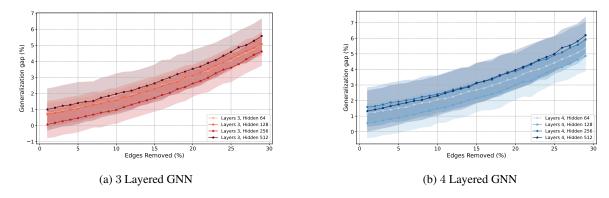
(a) 3 Layered GNN

(b) 4 Layered GNN

Figure 3: Generalization gap as a function of the percentage of perturbed edges values in node removal perturbation.

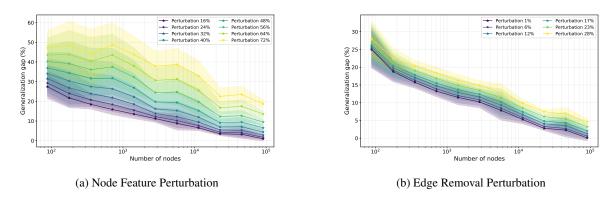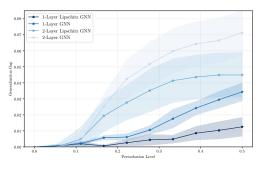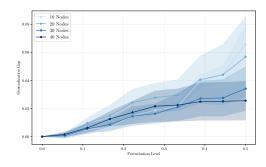(a) Node Feature Perturbation

(b) Edge Removal Perturbation

Figure 4: Generalization gap for edge and node perturbation for the Arxiv dataset for a 3 layered, 256 feature GNN.

node perturbations, we randomly zero out a percentage of the 128 dimensional embeddings for all the points in the dataset. Theoretically, that corresponds to modifying the underlying scalar function f (see equation (11)). For the edge perturbation, we randomly remove a percentage of the existing edges in the graph. This corresponds to perturbing the Laplace operator $\mathcal{L}$ (see equation (12)). We run these experiments for a varying number of nodes, by partitioning the training set in $\{1, 2, 4, 8, ...512, 1024\}$ partitions. We train the GNN with the *cross-entropy* loss, for 1000 epochs, using 0.005 learning rate, ADAM optimizer [24], and no weight decay with ReLU non-linearity. The purpose of the experiments is to show that GNN with more layers and more hidden units has a larger generalization gap under the same perturbation level. We further verify the relationship between the generalization gaps and the logarithm of the number of nodes in the graphs as indicated in Theorem 1.

In Figure 2 we can see the generalization gap as a function of the perturbation level for different GNN architectures. In these two figures, we confirm that GNNs with fewer layers and fewer hidden units are more robust to changes in the underlying manifold. This can be seen as the black lines (2 layers), are below both the red (3 layers) and the blue lines(4 layers). Intuitively, this can be attributed to overfitting the training set, given that more capacity in the GNN translates into a better training set accuracy. We also showcase that if the number of nodes in the training set is larger, the GNN is more robust to changes on the graph, given that the generalization gap increases more slowly in the GNN trained with 90941 nodes (Figure 2a), than in the one trained with 2841 nodes (Figure 2b). In Figure 3, we plot the generalization gap as a function of the perturbation magnitude for a GNN with 3 (Figure 3a), and 4 layers (Figure 3b). In both cases, the GNN with the largest number of hidden units is at the top, thus indicating that a larger generalization gap as indicated in Remark 1.

Figure 4 shows the generalization gaps of GNNs under node feature perturbations and edge perturbations decrease approximately linearly with the number of nodes in the logarithmic scale while increasing with the perturbation sizes. In Figure 4a and 4b, we show the generalization gap of a 3 layered GNN with 256 hidden units as a function of the number of nodes in the training set under node feature perturbation and edge removal perturbation, respectively. Each color in the figure represents a different perturbation level, going from less

(a) GNN with different architectures.

(b) GNN on different number of nodes.

Figure 5: Generalization gap as a function of number of nodes in the Point Cloud Classification.

perturbation (16% darker blue) to higher perturbation (72% lighter yellow). As can be seen in the plot, the GNNs generalization degrades as the perturbation level increases. Aligning with our theory, the generalization gap is linear with respect to the logarithm of the number of nodes. This is also true when the perturbation level increases.

Graph Classification with ModelNet Dataset

We evaluate the generalization gap of GNNs on graph level with the ModelNet10 dataset [61]. The dataset contains 3991 meshed CAD models from 10 categories for training and 908 models for testing. For each model, discrete points are uniformly randomly sampled from all points of the model to form the graphs. Each point is characterized by the 3D coordinates as features. Each node in the graph can be modeled as the sampling point and each edge weight is constructed based on the distance between each pair of nodes. The input graph features are set as the coordinates of each point, and the weights of the edges are calculated based on the Euclidean distance between the points. The mismatched point cloud model adds a Gaussian random variable with mean γ and variance 2γ to each coordinate of every sampled point. This can be seen as the underlying manifold mismatch. We calculate the generalization gap by training GNNs on graphs with $N = 40$ sampled points, and plotting the differences between the testing accuracy on the trained graphs sampled from normal point clouds and the testing accuracy on the graphs sampled from deformed point clouds with perturbation level γ. We implement Graph Neural Networks (GNN) with 1 and 2 layers with a single layer containing $F_0 = 3$ input features which are the 3d coordinates of each point, $F_1 = 64$ output features and $K = 5$ filter taps. While the architectures with 2 layers has another layer with $F_2 = 32$ features and 5 filter taps. We use the `ReLU` as nonlinearity. All architectures also include a linear readout layer mapping the final output features to a binary scalar that estimates the classification. Figure 5a shows the generalization gaps for GNNs with or without Lipschitz continuity assumptions of the graph filters. We observe that the continuity assumptions imposed on the graph filters help to improve the generalization capabilities as Theorem 2 shows. Figure 5b shows the results of GNNs on different number of nodes. We can see that the generalization gaps of GNNs scale with the size of the GNN architectures and scale with the size of mismatch. Figure 5b also shows that the generalization gap decreases with the number of nodes.

Conclusion

We showed that the robustness of GNN generalization to model mismatch from a manifold perspective. We focused on graphs derived from manifolds, and we established that GNNs equipped with low-pass and integral Lipschitz graph filters exhibit robust generalization. This generalization extends to previously unseen nodes and graphs derived from mismatched manifolds. Notably, we identified a tradeoff between the robust generalization and discriminability of GNNs. We validate our results both on node-level and graph-level classification tasks with real-world datasets.

References

[1] Eddie Aamari and Alexander Knop. Statistical query complexity of manifold estimation. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 116–122, 2021.

[2] Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. Graph convolution for semi-supervised classification: Improved linear separability and out-of-distribution generalization. In *International Conference on Machine Learning*, pages 684–693. PMLR, 2021.

[3] Monica Billio, Mila Getmansky, Andrew W Lo, and Loriana Pelizzon. Econometric measures of connectedness and systemic risk in the finance and insurance sectors. *Journal of financial economics*, 104(3):535–559, 2012.

[4] 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, 2017.

[5] Jeff Calder and Nicolas Garcia Trillos. Improved spectral convergence rates for graph laplacians on ε-graphs and k-nn graphs. *Applied and Computational Harmonic Analysis*, 60:123–175, 2022.

[6] Juan Cervino, Luana Ruiz, and Alejandro Ribeiro. Learning by transference: Training graph neural networks on growing graphs. *IEEE Transactions on Signal Processing*, 71:233–247, 2023.

[7] Rudrasis Chakraborty, Jose Bouza, Jonathan H Manton, and Baba C Vemuri. Manifoldnet: A deep neural network for manifold-valued data with applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(2):799–810, 2020.

[8] Joyce A Chew, Deanna Needell, and Michael Perlmutter. A convergence rate for manifold neural networks. In *2023 International Conference on Sampling Theory and Applications (SampTA)*, pages 1–5. IEEE, 2023.

[9] Heeyoul Choi and Seungjin Choi. Kernel isomap. *Electronics letters*, 40(25):1612–1613, 2004.

[10] Morris H. Degroot. Reaching a consensus. *Journal of the American Statistical Association*, 69(345):118–121, 1974.

[11] Mucong Ding, Kezhi Kong, Jiuhai Chen, John Kirchenbauer, Micah Goldblum, David Wipf, Furong Huang, and Tom Goldstein. A closer look at distribution shifts and out-of-distribution generalization on graphs. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

[12] Pascal Esser, Leena Chennuru Vankadara, and Debarghya Ghoshdastidar. Learning theory can (sometimes) explain generalisation in graph neural networks. *Advances in Neural Information Processing Systems*, 34:27043–27056, 2021.

[13] Shaohua Fan, Xiao Wang, Chuan Shi, Peng Cui, and Bai Wang. Generalizing graph neural networks on out-of-distribution graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[14] Amir Massoud Farahmand, Csaba Szepesvári, and Jean-Yves Audibert. Manifold-adaptive dimension estimation. In *Proceedings of the 24th international conference on Machine learning*, pages 265–272, 2007.

[15] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

[16] Fernando Gama, Antonio G Marques, Geert Leus, and Alejandro Ribeiro. Convolutional neural network architectures for signals supported on graphs. *IEEE Transactions on Signal Processing*, 67(4):1034–1049, 2019.

[17] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pages 3419–3430. PMLR, 2020.

[18] Walker Gosrich, Siddharth Mayya, Rebecca Li, James Paulos, Mark Yim, Alejandro Ribeiro, and Vijay Kumar. Coverage control in multi-robot systems via graph neural networks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8787–8793. IEEE, 2022.

[19] Alexander Grigor'yan. Heat kernels on weighted manifolds and applications. *Cont. Math*, 398(2006):93–191, 2006.

[20] Jiashu He, Charilaos I Kanatsoulis, and Alejandro Ribeiro. Network alignment with transferable graph autoencoders. *arXiv preprint arXiv:2310.03272*, 2023.

[21] Haotian Ju, Dongyue Li, Aneesh Sharma, and Hongyang R Zhang. Generalization in graph neural networks: Improved pac-bayesian bounds on graph diffusion. In *International Conference on Artificial Intelligence and Statistics*, pages 6314–6341. PMLR, 2023.

[22] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *In Mathematical Aspects of Deep Learning. Cambridge University Press*, 2022.

[23] Nicolas Keriven, Alberto Bietti, and Samuel Vaiter. Convergence and stability of graph convolutional networks on large random graphs. *Advances in Neural Information Processing Systems*, 33:21512–21523, 2020.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[26] Thien Le and Stefanie Jegelka. Limits, approximation and size transferability for gnns on sparse graphs via graphops. *Advances in Neural Information Processing Systems*, 36, 2024.

[27] Jaekoo Lee, Hyunjae Kim, Jongsun Lee, and Sungroh Yoon. Transfer learning for deep learning on graph-structured data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[28] Ron Levie. A graphon-signal analysis of graph neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

[29] 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.

[30] Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. Ood-gnn: Out-of-distribution generalized graph neural network. *IEEE Transactions on Knowledge and Data Engineering*, 35(7):7328–7340, 2022.

[31] Renjie Liao, Raquel Urtasun, and Richard Zemel. A pac-bayesian approach to generalization bounds for graph neural networks. In *International Conference on Learning Representations*, 2020.

[32] László Lovász. *Large networks and graph limits*, volume 60. American Mathematical Soc., 2012.

[33] Dalton Lunga, Saurabh Prasad, Melba M Crawford, and Okan Ersoy. Manifold-learning-based feature extraction for classification of hyperspectral data: A review of advances in manifold learning. *IEEE Signal Processing Magazine*, 31(1):55–66, 2013.

[34] Jiaqi Ma, Junwei Deng, and Qiaozhu Mei. Subgroup generalization and fairness of graph neural networks. *Advances in Neural Information Processing Systems*, 34:1048–1061, 2021.

[35] Sohir Maskey, Gitta Kutyniok, and Ron Levie. Generalization bounds for message passing networks on mixture of graphons. *arXiv preprint arXiv:2404.03473*, 2024.

[36] Sohir Maskey, Ron Levie, and Gitta Kutyniok. Transferability of graph neural networks: an extended graphon approach. *Applied and Computational Harmonic Analysis*, 63:48–83, 2023.

[37] Sohir Maskey, Ron Levie, Yunseok Lee, and Gitta Kutyniok. Generalization analysis of message passing neural networks on large random graphs. *Advances in neural information processing systems*, 35:4805–4817, 2022.

[38] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[39] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

[40] Hariharan Narayanan and Partha Niyogi. On the sample complexity of learning smooth cuts on a manifold. In *COLT*, 2009.

[41] Alejandro Parada-Mayorga, Zhiyang Wang, and Alejandro Ribeiro. Graphon pooling for reducing dimensionality of signals and convolutional operators on graphs. *IEEE Transactions on Signal Processing*, 2023.

[42] Raksha Ramakrishna, Hoi-To Wai, and Anna Scaglione. A user guide to low-pass graph signal processing and its applications: Tools and applications. *IEEE Signal Processing Magazine*, 37(6):74–85, 2020.

[43] Luana Ruiz, Luiz Chamon, and Alejandro Ribeiro. Graphon neural networks and the transferability of graph neural networks. *Advances in Neural Information Processing Systems*, 33:1702–1712, 2020.

[44] Luana Ruiz, Luiz FO Chamon, and Alejandro Ribeiro. Transferability properties of graph neural networks. *IEEE Transactions on Signal Processing*, 2023.

[45] Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61(7):1644–1656, 2013.

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

[47] Yangyi Shen, Beatrice Bevilacqua, Joshua Robinson, Charilaos Kanatsoulis, Jure Leskovec, and Bruno Ribeiro. Zero-shot generalization of gnns over distinct attribute domains. In *ICML 2024 Workshop on Theoretical Foundations of Foundation Models*, 2024.

[48] Ameet Talwalkar, Sanjiv Kumar, and Henry Rowley. Large-scale manifold learning. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[49] Qin Tian, Wenjun Wang, Chen Zhao, Minglai Shao, Wang Zhang, and Dong Li. Graphs generalization under distribution shifts. *arXiv preprint arXiv:2403.16334*, 2024.

[50] Loring W.. Tu. *An introduction to manifolds*. Springer., 2011.

[51] 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.

[52] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 2020.

[53] Zhiyang Wang, Juan Cervino, and Alejandro Ribeiro. A manifold perspective on the statistical generalization of graph neural networks. *arXiv preprint arXiv:2406.05225*, 2024.

[54] Zhiyang Wang, Juan Cerviño, and Alejandro Ribeiro. Generalization of geometric graph neural networks. *submitted to Asilomar Conference on Signals, Systems, and Computers 2024.*, 2024.

[55] Zhiyang Wang, Juan Cerviño, and Alejandro Ribeiro. A manifold perspective on the statistical generalization of graph neural networks. *arXiv preprint arXiv:2406.05225*, 2024.

[56] Zhiyang Wang, Luana Ruiz, and Alejandro Ribeiro. Convolutional neural networks on manifolds: From graphs and back. In *2022 56th Asilomar Conference on Signals, Systems, and Computers*, pages 356–360. IEEE, 2022.

[57] Zhiyang Wang, Luana Ruiz, and Alejandro Ribeiro. Geometric graph filters and neural networks: Limit properties and discriminability trade-offs. *IEEE Transactions on Signal Processing*, 2024.

[58] Zhiyang Wang, Luana Ruiz, and Alejandro Ribeiro. Stability to deformations of manifold filters and manifold neural networks. *IEEE Transactions on Signal Processing*, pages 1–15, 2024.

[59] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

[60] Yiming Wu and Kap Luk Chan. An extended isomap algorithm for learning multi-class manifold. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826)*, volume 6, pages 3429–3433. IEEE, 2004.

[61] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[62] Bo Yang, Ming Xiang, and Yupei Zhang. Multi-manifold discriminant isomap for visualization and classification. *Pattern Recognition*, 55:215–230, 2016.

[63] Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, pages 11975–11986. PMLR, 2021.

[64] Nan Yin, Li Shen, Mengzhu Wang, Long Lan, Zeyu Ma, Chong Chen, Xian-Sheng Hua, and Xiao Luo. Coco: A coupled contrastive framework for unsupervised domain adaptive graph classification. In *International Conference on Machine Learning*, pages 40040–40053. PMLR, 2023.

[65] Xianchen Zhou and Hongxia Wang. The generalization error of graph convolutional networks may enlarge with more layers. *Neurocomputing*, 424:97–106, 2021.

[66] Qi Zhu, Carl Yang, Yidan Xu, Haonan Wang, Chao Zhang, and Jiawei Han. Transfer learning of graph neural networks with ego-graph information maximization. *Advances in Neural Information Processing Systems*, 34:1766–1779, 2021.

Figure 6: Point cloud models in ModelNet10 with $N = 300$ sampled points in each model, corresponding to bathtub, chair, desk, table, toiler, and bed.

Experiment Details

All experiments were done with a *NVIDIA GeForce RTX 3090*, and each set of experiments took at most 10 hours to complete. The ArXiv dataset and ModelNet10 dataset used in this paper are public and free to use. They can be downloaded using the *pytorch* package (https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html), the *ogb* package (https://ogb.stanford.edu/docs/nodeprop/) and the Princeton ModelNet project (https://modelnet.cs.princeton.edu/). In total, the datasets occupy around 5 GB. However, they do not need to be all stored at the same time, as the experiments that we run can be done in series. We will release the codes after the paper is accepted.

Node Classification

we used the graph convolutional layer `GCN`, and trained for 1000 epochs. For the optimizer, we used `AdamW`, with using a learning rate of 0.01, and 0 weight decay. We trained using the graph convolutional layer, with a varying number of layers and hidden units. For dropout, we used 0.5. We trained using the cross-entropy loss. In all cases, we trained 2 and 3 layered GNNs.

ModelNet10 graph classification tasks

ModelNet10 dataset [61] includes 3,991 meshed CAD models from 10 categories for training and 908 models for testing as Figure 6 shows. In each model, N points are uniformly randomly selected to construct graphs to approximate the underlying model, such as chairs, tables.

The weight function of the constructed graph is determined as (6) with $\epsilon = 0.001$. We calculate the Laplacian matrix for each graph as the input graph shift operator. In this experiment, we implement GNNs with different numbers of layers and hidden units with $K = 5$ filters in each layer. All the GNN architectures are trained by minimizing the cross-entropy loss. We implement an ADAM optimizer with the learning rate set as 0.005 along with the forgetting factors 0.9 and 0.999. We carry out the training for 40 epochs with the size of batches set as 10. We run 5 random dataset partitions and show the average performances and the standard deviation across these partitions.

Low Pass Filter Assumption

In the main results, we assume that the GNN and MNN are low-pass filters. We believe this is a mild assumption. A high frequency on the graph/manifold implies that there exist signals (eigenvectors/functions) that can have large variations in adjacent entries. This naturally generates instabilities, which are more difficult to learn. It is expected certain amount of homogeneity within the neighborhood of a node, translates into low-frequency signals. It has been shown in practice that several real-world physical models lead to low-pass filters such as opinion dynamics [10], econometrics [3], and graph signal processing [42].

Several other learning techniques rely on low-pass filters. Principal Component Analysis (PCA), can be seen as low-pass filters applied to the correlation matrix. Isomap, is another example of low pass filtering, but in this case applied to the Laplacian matrix. In all, we believe that the low pass filter assumption is not a hard assumption, given its empirical verification, and the fact that several other techniques that work in practice utilize it.

Proofs

Proof of Theorem 1

The generalization error of GNNs under deformations is defined as

$$GA_\tau = \sup_{\mathbf{H} \in \mathcal{H}} |R_{\mathcal{M}^\tau}(\mathbf{H}) - R_{\mathbf{G}}(\mathbf{H})| . \tag{22}$$

An intermediate term $\int_\mathcal{M} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(x)\right) \mathrm{d}\mu(x)$ is first subtracted and added as follows.

$$|R_{\mathcal{M}^\tau}(\mathbf{H}) - R_{\mathbf{G}}(\mathbf{H})| = \left| \frac{1}{N} \sum_{i=1}^N \ell([\boldsymbol{\Phi}(\mathbf{H}, \mathbf{L}, \mathbf{x})]_i, [\mathbf{y}]_i) - \int_{\mathcal{M}^\tau} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}_\tau, f)(x), g(x)\right) \mathrm{d}\mu_\tau(x) \right| \tag{23}$$

$$= \left| \frac{1}{N} \sum_{i=1}^N \ell([\boldsymbol{\Phi}(\mathbf{H}, \mathbf{L}, \mathbf{x})]_i, [\mathbf{y}]_i) - \int_\mathcal{M} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(x)\right) \mathrm{d}\mu(x) \right.$$

$$\left. + \int_\mathcal{M} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(x)\right) \mathrm{d}\mu(x) - \int_{\mathcal{M}^\tau} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}_\tau, f)(x), g(x)\right) \mathrm{d}\mu_\tau(x) \right| \tag{24}$$

This can be further decomposed with absolute value inequality as follows.

$$|R_{\mathcal{M}^\tau}(\mathbf{H}) - R_{\mathbf{G}}(\mathbf{H})| \leq \left| \frac{1}{N} \sum_{i=1}^N \ell([\boldsymbol{\Phi}(\mathbf{H}, \mathbf{L}, \mathbf{x})]_i, [\mathbf{y}]_i) - \int_\mathcal{M} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(x)\right) \mathrm{d}\mu(x) \right| \tag{25}$$

$$+ \left| \int_\mathcal{M} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(x)\right) \mathrm{d}\mu(x) - \int_{\mathcal{M}^\tau} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}_\tau, f)(x), g(x)\right) \mathrm{d}\mu_\tau(x) \right|$$

The first part in (26) can be found in Theorem 1 of [55] with the proof in Appendix D. With the conclusion written as

$$\left| \frac{1}{N} \sum_{i=1}^N \ell([\boldsymbol{\Phi}(\mathbf{H}, \mathbf{L}, \mathbf{x})]_i, [\mathbf{y}]_i) - \int_\mathcal{M} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(x)\right) \mathrm{d}\mu(x) \right|$$

$$\leq C_1 \frac{\epsilon}{\sqrt{N}} + C_2 \frac{\sqrt{\log(1/\delta)}}{N} + C_3 \left(\frac{\log N}{N} \right)^{\frac{1}{d}}, \tag{26}$$

with $C_1 = C_{\mathcal{M},1} \frac{\pi^2}{6} \|f\|_\mathcal{M} + C_{\mathcal{M},2} \frac{\pi^2}{6}$. $C_{\mathcal{M},1}$, $C_{\mathcal{M},2}$ and C_3 depend on d and the volume, the probability density, the injectivity radius and sectional curvature of $\mathcal{M}$, $C_2 = \frac{\pi^2}{6}$.

We focus on the integration on the mismatched manifold first.

$$\int_{\mathcal{M}^\tau} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}_\tau, f)(x), g(x)\right) \mathrm{d}\mu_\tau(x) = \int_\mathcal{M} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}, f)(\tau(x)), g(\tau(x))\right) \mathrm{d}\mu(\tau(x)) \tag{27}$$

$$= \int_\mathcal{M} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}', f)(x), g(\tau(x))\right) J_x(\tau) \mathrm{d}\mu(x), \tag{28}$$

with $\mathcal{L}' f(x) = \mathcal{L} f(\tau(x))$. The second part of (26) therefore can be written as

$$\left| \int_\mathcal{M} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(x)\right) \mathrm{d}\mu(x) - \int_{\mathcal{M}^\tau} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}_\tau, f)(x), g(x)\right) \mathrm{d}\mu_\tau(x) \right|$$

$$= \left| \int_\mathcal{M} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(x)\right) \mathrm{d}\mu(x) - \int_\mathcal{M} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}', f)(x), g(\tau(x))\right) J_x(\tau) \mathrm{d}\mu(x) \right| \tag{29}$$

By subtracting and adding an intermediate term $\int_\mathcal{M} \ell\left(\boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(\tau(x))\right) \mathrm{d}\mu(x)$ and with the absolute

value inequality, we have

$$\left| \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(x)\right) \mathrm{d}\mu(x) - \int_{\mathcal{M}^\tau} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}_\tau, f)(x), g(x)\right) \mathrm{d}\mu_\tau(x) \right|$$

$$\leq \left| \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(x)\right) \mathrm{d}\mu(x) - \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(\tau(x))\right) \mathrm{d}\mu(x) \right|$$

$$+ \left| \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(\tau(x))\right) \mathrm{d}\mu(x) - \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}', f)(x), g(\tau(x))\right) J_x(\tau) \mathrm{d}\mu(x) \right| \quad (30)$$

The first part of (30) can be bounded by the Lipschitz continuity of loss function ℓ, which leads to

$$\left| \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(x)\right) \mathrm{d}\mu(x) - \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(\tau(x))\right) \mathrm{d}\mu(x) \right|$$

$$\leq \int_{\mathcal{M}} \left| \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(x)\right) - \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(\tau(x))\right) \right| \mathrm{d}\mu(x) \leq \int_{\mathcal{M}} |g(x) - g(\tau(x))| \mathrm{d}\mu(x). \quad (31)$$

With the Lipschitz continuity of target function g and the bounded distance between x and $\tau(x)$ as $dist(x, \tau(x)) \leq \gamma$, we have

$$\int_{\mathcal{M}} |g(x) - g(\tau(x))| \mathrm{d}\mu(x) \leq C_g \gamma. \quad (32)$$

Therefore, the first part of (30) can be bounded as follows.

$$\left| \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(x)\right) \mathrm{d}\mu(x) - \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(\tau(x))\right) \mathrm{d}\mu(x) \right| \leq C_g \gamma. \quad (33)$$

The second part of (30) can be first bounded by decomposing the Jacobian matrix as

$$\left| \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(\tau(x))\right) \mathrm{d}\mu(x) - \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}', f)(x), g(\tau(x))\right) J_x(\tau) \mathrm{d}\mu(x) \right|$$

$$= \left| \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(\tau(x))\right) \mathrm{d}\mu(x) - \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}', f)(x), g(\tau(x))\right) (1 + \Delta_\gamma) \mathrm{d}\mu(x) \right|$$

$$\leq \left| \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(\tau(x))\right) \mathrm{d}\mu(x) - \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}', f)(x), g(\tau(x))\right) \mathrm{d}\mu(x) \right|$$

$$+ \|\Delta_\gamma\|_F \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}', f)(x), g(\tau(x))\right) \mathrm{d}\mu(x) \quad (34)$$

$$\leq \left| \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(\tau(x))\right) \mathrm{d}\mu(x) - \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}', f)(x), g(\tau(x))\right) \mathrm{d}\mu(x) \right|$$

$$+ \gamma \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}', f)(x), g(\tau(x))\right) \mathrm{d}\mu(x), \quad (35)$$

where the last step is bounded by the assumption that $\|J_x(\tau) - I\|_F \leq \gamma$. With the Lipschitz continuity and the boundedness of loss function ℓ, we have

$$\left| \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x), g(\tau(x))\right) \mathrm{d}\mu(x) - \int_{\mathcal{M}} \ell\left(\mathbf{\Phi}(\mathbf{H}, \mathcal{L}', f)(x), g(\tau(x))\right) J_x(\tau) \mathrm{d}\mu(x) \right|$$

$$\leq \int_{\mathcal{M}} |\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)(x) - \mathbf{\Phi}(\mathbf{H}, \mathcal{L}', f)(x)| \mathrm{d}\mu(x) + \gamma \ell_{max} \quad (36)$$

$$= \|\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f) - \mathbf{\Phi}(\mathbf{H}, \mathcal{L}', f)\|_1 + \gamma \ell_{max} \quad (37)$$

$$\leq \sqrt{d} \|\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f) - \mathbf{\Phi}(\mathbf{H}, \mathcal{L}', f)\|_2 + \gamma \ell_{max}, \quad (38)$$

16

where the last inequality comes from the fact that $\|f\|_1 \leq \sqrt{d}\|f\|_2$ for $f \in L^2(\mathcal{M})$. The loss function value is bounded as the output of MNN and the target function are both bounded. To study the difference of the MNN outputs, we import the stability property of manifold neural networks in [58]. Specifically, we utilize Theorem 1, which interprets the manifold deformation as perturbations of Laplacian operator.

Theorem 3 (Theorem 1 in [58]). *Let $\mathcal{L}$ be the LB operator of manifold $\mathcal{M}$. Let $\tau : \mathcal{M} \to \mathcal{M}$ be a manifold perturbation such that $\text{dist}(x, \tau(x)) \leq \gamma$ and $\|J_x(\tau) - I\|_F \leq \gamma$ for all $x \in \mathcal{M}$. If the gradient field is smooth, it holds that*

$$\mathcal{L} - \mathcal{L}' = \mathbf{E}\mathcal{L} + \mathcal{A}, \tag{39}$$

where $\mathbf{E}$ and $\mathcal{A}$ satisfy $\|\mathbf{E}\| = O(\gamma)$ and $\|\mathcal{A}\|_{op} = O(\gamma)$.

We further import Lemma 1, Lemma 2 and Lemma 3 from [58].

Proposition 1 (Lemma 1 and Lemma 3 in [58]). *The eigenvalues of LB operators $\mathcal{L}$ and perturbed $\mathcal{L}' = \mathcal{L} + \mathbf{E}\mathcal{L} + \mathcal{A}$, with $\|\mathbf{E}\| = O(\gamma)$ and $\|\mathcal{A}\|_{op} = O(\gamma)$ satisfy*

$$|\lambda_i - \lambda_i'| \leq \gamma|\lambda_i| + \gamma, \text{ for all } i = 1, 2 \ldots \tag{40}$$

Let $\beta = \min\{\lambda_{M+1} - \lambda_M, \gamma\lambda_2 + \gamma\}$ with $M = \max\{i|\lambda_i \leq \lambda\}$. We denote the set of eigenvalue indices with the difference compared to the previous eigenvalue larger than γ as

$$I = \{1\} \cup \{i > 1|\lambda_i - \lambda_{i-1} \geq \beta\}, \tag{41}$$

with $|I| = K$. We define a new eigenvalue sequence with

$$J_i = \begin{cases} i - 1 & \text{if } I_k \leq i < I_{k+1}, k = 1, 2 \cdots, K \\ i & \text{otherwise} \end{cases} \tag{42}$$

We start with the difference of manifold filters by first writing out the spectral representation as

$$\mathbf{h}(\mathcal{L})f = \sum_{i=1}^{M} \hat{h}(\lambda_i)\langle f, \boldsymbol{\phi}_i\rangle\boldsymbol{\phi}_i. \tag{43}$$

The difference can be written as

$$|\mathbf{h}(\mathcal{L})f - \mathbf{h}(\mathcal{L}')f| = \left|\sum_{i=1}^{M} \hat{h}(\lambda_i)\langle f, \boldsymbol{\phi}_i\rangle\boldsymbol{\phi}_i - \sum_{i=1}^{M} \hat{h}(\lambda_i')\langle f, \boldsymbol{\phi}_i'\rangle\boldsymbol{\phi}_i'\right| \tag{44}$$

We subtract and add an intermediate term $\sum_{i=1}^{M} \hat{h}(\lambda_{J_i})\langle f, \boldsymbol{\phi}_i\rangle\boldsymbol{\phi}_i$. With the absolute value inequality, we can decompose the difference as

$$\|\mathbf{h}(\mathcal{L})f - \mathbf{h}(\mathcal{L}')f\|$$
$$= \left\|\sum_{i=1}^{M} \hat{h}(\lambda_i)\langle f, \boldsymbol{\phi}_i\rangle\boldsymbol{\phi}_i - \sum_{i=1}^{M} \hat{h}(\lambda_{J_i})\langle f, \boldsymbol{\phi}_i\rangle\boldsymbol{\phi}_i + \sum_{i=1}^{M} \hat{h}(\lambda_{J_i})\langle f, \boldsymbol{\phi}_i\rangle\boldsymbol{\phi}_i - \sum_{i=1}^{M} \hat{h}(\lambda_i')\langle f, \boldsymbol{\phi}_i'\rangle\boldsymbol{\phi}_i'\right\| \tag{45}$$
$$\leq \left\|\sum_{i=1}^{M} \hat{h}(\lambda_i)\langle f, \boldsymbol{\phi}_i\rangle\boldsymbol{\phi}_i - \sum_{i=1}^{M} \hat{h}(\lambda_{J_i})\langle f, \boldsymbol{\phi}_i\rangle\boldsymbol{\phi}_i\right\| + \left\|\sum_{i=1}^{M} \hat{h}(\lambda_{J_i})\langle f, \boldsymbol{\phi}_i\rangle\boldsymbol{\phi}_i - \sum_{i=1}^{M} \hat{h}(\lambda_i')\langle f, \boldsymbol{\phi}_i'\rangle\boldsymbol{\phi}_i'\right\|. \tag{46}$$

The first term in (46) can be decomposed by subtracting and adding an intermediate term $\sum_{i=1}^{M} \hat{h}(\lambda_{J_i})\langle f, \boldsymbol{\phi}_i\rangle\boldsymbol{\phi}_i$. With absolute value inequality, we have

$$\left\|\sum_{i=1}^{M} \hat{h}(\lambda_i)\langle f, \boldsymbol{\phi}_i\rangle\boldsymbol{\phi}_i - \sum_{i=1}^{M} \hat{h}(\lambda_{J_i})\langle f, \boldsymbol{\phi}_i\rangle\boldsymbol{\phi}_i\right\|$$
$$\leq \sum_{i=1}^{M} \left|\hat{h}(\lambda_i) - \hat{h}(\lambda_{J_i})\right| \|f\|_{\mathcal{M}}\|\boldsymbol{\phi}_i\|^2 \tag{47}$$
$$\leq \sum_{i=1}^{M} \left|\hat{h}'(\lambda_i)\right| |\lambda_i - \lambda_{J_i}|\|f\|_{\mathcal{M}} \tag{48}$$
$$\leq \sum_{i=1}^{M} C_L\lambda_i^{-d-1}\beta\|f\|_{\mathcal{M}} \leq \frac{\pi^2}{6}C_L\beta\|f\|_{\mathcal{M}}. \tag{49}$$

As M becomes very large, Weyl's law which indicates that eigenvalues of Laplace operator scales with the order $\lambda_i \sim i^{2/d}$. Then $\sum_{i=1}^{M} \lambda_i^{-d-1}$ is in the order $\sum_{i=1}^{M} i^{-2}$, which is bounded by $\frac{\pi^2}{6}$.

The second term in (46) can be rewritten with the definition of J_i and I as

$$
\left\| \sum_{i=1}^{M} \hat{h}(\lambda_{J_i}) \langle f, \phi_i \rangle \phi_i - \sum_{i=1}^{M} \hat{h}(\lambda_i') \langle f, \phi_i' \rangle \phi_i' \right\|
$$

$$
= \left\| \sum_{i=1}^{M} \hat{h}(\lambda_{J_i}) \langle f, \phi_i \rangle \phi_i - \sum_{i=1}^{M} \hat{h}(\lambda_{J_i}) \langle f, \phi_i' \rangle \phi_i' + \sum_{i=1}^{M} \hat{h}(\lambda_{J_i}) \langle f, \phi_i' \rangle \phi_i' - \sum_{i=1}^{M} \hat{h}(\lambda_i') \langle f, \phi_i' \rangle \phi_i' \right\| \tag{50}
$$

$$
\leq \left\| \sum_{i=1}^{M} \hat{h}(\lambda_{J_i}) \langle f, \phi_i \rangle \phi_i - \sum_{i=1}^{M} \hat{h}(\lambda_{J_i}) \langle f, \phi_i' \rangle \phi_i' \right\| + \left\| \sum_{i=1}^{M} \hat{h}(\lambda_{J_i}) \langle f, \phi_i' \rangle \phi_i' - \sum_{i=1}^{M} \hat{h}(\lambda_i') \langle f, \phi_i' \rangle \phi_i' \right\| \tag{51}
$$

The first term in (51) can be bounded as

$$
\left\| \sum_{i=1}^{M} \hat{h}(\lambda_{J_i}) \langle f, \phi_i \rangle \phi_i - \sum_{i=1}^{M} \hat{h}(\lambda_{J_i}) \langle f, \phi_i' \rangle \phi_i' \right\|
$$

$$
\leq \left\| \sum_{k=1}^{K} \lambda_{I_k}^{-d} \sum_{I_k \leq i < I_{k+1}} \langle f, \phi_i \rangle \phi_i - \sum_{k=1}^{K} \lambda_{I_{k+1}}^{-d} \sum_{I_k \leq i < I_{k+1}} \langle f, \phi_i' \rangle \phi_i' \right\| \tag{52}
$$

$$
\leq \left\| \sum_{k=1}^{K} \lambda_{I_k}^{-d} F_k - \sum_{k=1}^{K} \lambda_{I_{k+1}}^{-d} F_k' \right\| \tag{53}
$$

$$
= \left\| \sum_{k=1}^{K} \lambda_{I_k}^{-d} \langle f, \mathbf{\Phi}_k \rangle \mathbf{\Phi}_k - \sum_{k=1}^{K} \lambda_{I_{k+1}}^{-d} \langle f, \mathbf{\Phi}_k' \rangle \mathbf{\Phi}_k' \right\| \tag{54}
$$

$$
\leq 2 \sum_{k=1}^{K} \lambda_{I_k}^{-d} \|f\|_{\mathcal{M}} \|\mathbf{\Phi}_k - \mathbf{\Phi}_k'\| \tag{55}
$$

$$
\leq \frac{\pi^3}{3} \frac{\gamma}{|\lambda_{M+1} - \lambda_M|} \|f\|_{\mathcal{M}}, \tag{56}
$$

where (56) is derived based on Lemma 2 in [58], which is Davis-Khan theorem. Similarly, as K becomes large, consider $\lambda_i \sim i^{2/d}$, the summation can be bounded as $\frac{\pi^2}{6}$. We denote the eigenvalue gap as $\theta^{-1} = |\lambda_{M+1} - \lambda_M|^{-1}$, which is finite as M is always finite even as the number of nodes grow.

The second term in (51) can be bounded similarly as steps in (47) to (49).

$$
\left\| \sum_{i=1}^{M} \hat{h}(\lambda_{J_i}) \langle f, \phi_i' \rangle \phi_i' - \sum_{i=1}^{M} \hat{h}(\lambda_i') \langle f, \phi_i' \rangle \phi_i' \right\| \leq \left(\frac{\pi^2}{6} \beta + \frac{\pi^2}{3} \gamma \right) C_L \|f\|_{\mathcal{M}}. \tag{57}
$$

Combine (38), (49) and (56) together with (57), we can get the bound for the second term of (26), which leads to $C_4 = C_g + \ell_{max} + \frac{\pi^2(\lambda_2+2)}{3} \sqrt{d} C_L \|f\|_{\mathcal{M}} + \frac{\pi^3}{3} \theta^{-1} \|f\|_{\mathcal{M}}$.

This concludes that

$$
GA_\tau \leq C_1 \frac{\epsilon}{\sqrt{N}} + C_2 \frac{\sqrt{\log(1/\delta)}}{N} + C_3 \left(\frac{\log N}{N} \right)^{\frac{1}{d}} + C_4 \gamma. \tag{58}
$$

Proof of Theorem 2

Theorem 2 can be extended based on the proof process of Theorem 1. Except there are K manifolds to consider parallelly, which leads to a summation of all the K manifold generalization errors. The only difference lies in that the target function remains the same even under model mismatch, which leads to the constant C_4 independent of the target function.

Extension to Out-of-Distribution Generalization

The authors of [47] propose a novel method for mapping the features from different domains. In [11], the authors show that GNNs with augmentations perform better under domain shifts with pervasive empirical results. The

18

authors in [13] provide a casual representational learning method for GNN out-of-distribution generalization. These methods and architectures to cope with domain shifts are summarized in [30]. While only the authors in [2] provide a theoretical analysis on out-of-distribution generalization with a contextual stochastic block model as the graph generative model.

An important implication of generalization over deformation is to characterize the domain generalization capability. The conclusion of GNN generalization with model mismatch on the node level can extend to bound the domain generalization gap of GNNs by seeing the domain shift as the manifold model shift.

Suppose the nodes on the training graph $\mathbf{G}_1$ are sampled from manifold $\mathcal{M}_1$ (equipped with Laplace operator $\mathcal{L}_1$) while the testing nodes are sampled from manifold $\mathcal{M}_2$ (equipped with Laplace operator $\mathcal{L}_2$). The out-of-distribution generalization error is denoted as

$$GA_{OOD} = \sup_{\mathbf{H} \in \mathcal{H}} |R_{\mathcal{M}_2}(\mathbf{H}) - R_{\mathbf{G}_1}(\mathbf{H})|, \tag{59}$$

with $R_{\mathcal{M}_2}$ representing the statistical risk over the shifted manifold $\mathcal{M}_2$ and $R_{\mathbf{G}_1}$ the empirical risk over the training graph $\mathbf{G}_1$ sampled from $\mathcal{M}_1$. A similar conclusion can be obtained by measuring the operator difference between $\mathcal{L}_1$ and $\mathcal{L}_2$, which is used to measure the difference between models $\mathcal{M}_1$ and $\mathcal{M}_2$.

Proposition 2. *Suppose a neural network is equipped with filters defined in Definition 2 and normalized non-linearites (Assumption 2). The neural network is operated on a graph $\mathbf{G}_1$ generated according to (6) from $\mathcal{M}_1$ and a shifted manifold $\mathcal{M}_2$ with a bandlimited (Definition 1) input manifold function. The out-of-distribution generalization of neural network trained on $\mathbf{G}_1$ to minimize a normalized Lipschitz loss function (Assumption 3) holds in probability at least $1 - \delta$ that*

$$GA_{OOD} \leq C_1 \frac{\epsilon}{\sqrt{N}} + C_2 \frac{\sqrt{\log(1/\delta)}}{N} + C_3 \left(\frac{\log N}{N} \right)^{\frac{1}{d}} + C_4 \|\mathcal{L}_1 - \mathcal{L}_2\|_{op},$$

with C_1 and C_3 depending on the geometry of $\mathcal{M}_1$, C_4 depending on the Lipschitz continuity of the target function, C_1 and C_4 depending on the continuity of the filter functions.

Proof. The proof process is in accordance with the proof of Theorem 1, except that the differences between eigenvalues of $\mathcal{M}_1$ and $\mathcal{M}_2$ are now bounded as follows.

Lemma 1. *The eigenvalues of LB operators $\mathcal{L}_1$ and $\mathcal{L}_2$ satisfy*

$$|\lambda_{1,i} - \lambda_{2,i}| \leq \|\mathcal{L}_1 - \mathcal{L}_2\|_{op}, \text{ for all } i = 1, 2 \ldots \tag{60}$$

Combined with the Davis-Khan Theorem to bound the eigenspace difference as

Lemma 2. *Suppose the spectra of $\mathcal{L}_1$ and $\mathcal{L}_2$ are partitioned as $\theta \bigcup \Theta$ and $\omega \bigcup \Omega$ respectively, with $\theta \bigcap \Omega = \emptyset$ and $\omega \bigcap \Theta = \emptyset$. If there exists $\beta > 0$ such that $\min_{x \in \theta, y \in \Omega} |x - y| \geq \beta$ and $\min_{x \in \omega, y \in \Theta} |x - y| \geq \beta$, then*

$$\|E_{\mathcal{L}_1}(\theta) - E_{\mathcal{L}_2}(\omega)\| \leq \frac{\pi}{2} \frac{\|\mathcal{L}_1 - \mathcal{L}_2\|_{op}}{d} \tag{61}$$

$\square$

Further Reference

Graphon theory   Some works consider the generative model of graphs as graphons. These works covered the properties of GNNs, including convergence, stability, and transferability [23, 36, 43]. Graphon can also be used as a pooling operator in GNNs [41]. Although graphon is a straightforward model with simple calculations, it imposes several limitations compared to the manifold model that we prefer. Firstly, the graphon model assumes an infinite degree at every node of the generative graph [32], which is not the case in most scenarios in practice. Secondly, the graphon model gives little intuition about the geometry of the underlying model, which makes it hard to visualize a graphon. On the other hand, manifolds are more interpretable, as simple cases can be found on the sphere or other 3D shapes. Lastly, the manifold model is a more realistic assumption even for high-dimensional data [33].

Manifold Theory   We consider a manifold model to be a generative model of graphs. Related to us, some works have focused on manifold learning. In particular, some works have set out to test the manifold assumption of the data [15], the manifold dimension [14] and show the complexity of doing so [1, 40]. In terms of prediction and classification using manifolds and manifold data, several algorithms and methods have been proposed – notable the *Isomap* algorithm [9, 60, 62] and other manifold learning techniques [48]. In all, these techniques utilize the manifold assumption to infer properties of the manifold, which differs from our goal of providing a generalization bound for GNNs generated from the manifold.

Transferability of GNNs   Transferability of GNNs has been analyzed in many works by comparing the output differences of GNNs on different sizes of graphs when graphs converge to a limit model without statistical generalization analysis. In [36, 43, 44], the transferability of GNNs on graphs generated from graphon models is analyzed by bounding the output differences of GNNs. In [6], the authors show how increasing the size of the graph as the GNN learns, generalizes to the large-scale graph. In [29], the authors analyze the transferability of GNNs on graphs generated from a general topological space while the authors in [57] focus on graphs generated from manifolds. In [26], the authors propose a novel graphop operator as a limit model for both dense and sparse graphs with a transferability result proved. In [27] and [66], the authors study the transfer learning of GNNs with measurements of distance between graphs without a limit assumption. In [20], a transferable graph transformer is proposed and proved with empirical results.