Stability Properties of Graph Neural Networks

Fernando Gama, Joan Bruna, and Alejandro Ribeiro

***Abstract*—Graph neural networks (GNNs) have emerged as a powerful tool for nonlinear processing of graph signals, exhibiting success in recommender systems, power outage prediction, and motion planning, among others. GNNs consist of a cascade of layers, each of which applies a graph convolution, followed by a pointwise nonlinearity. In this work, we study the impact that changes in the underlying topology have on the output of the GNN. First, we show that GNNs are permutation equivariant, which implies that they effectively exploit internal symmetries of the underlying topology. Then, we prove that graph convolutions with integral Lipschitz filters, in combination with the frequency mixing effect of the corresponding nonlinearities, yields an architecture that is both stable to small changes in the underlying topology, and discriminative of information located at high frequencies. These are two properties that cannot simultaneously hold when using only linear graph filters, which are either discriminative or stable, thus explaining the superior performance of GNNs.**

***Index Terms*—Graph convolutions, graph filters, graph neural networks, graph signal processing, network data, stability.**

I. Introduction

CONVOLUTIONAL neural networks (CNNs) [1] are the tool of choice for machine learning in Euclidean space. CNNs consist of nonlinear maps in which the output follows from sending the input through a cascade of layers, each of which computes a convolution with a bank of filters followed by a pointwise nonlinearity [2, Ch. 9]. The value of the filter taps in the convolution is obtained by minimizing some cost function over a training set. The success of CNNs is simultaneously predictable and surprising. If we were to restrict attention to linear machine learning, a century of empirical and theoretical evidence would prescribe the use of convolutional filters. It is then predictable that the addition of a pointwise nonlinearity, which on the face of it is a pretty minor modification, is a sensible choice for a nonlinear processing architecture. But at the same time it is surprising that such a minor modification produces so much of an effect on empirically observed performance.

An answer to this question was put forth in [3] in the form of stability to diffeomorphisms. This seminal work considers *scattering transforms* which are information processing architectures akin to CNNs. They are also built from convolutions and nonlinearities but use pre-specified families of wavelet frames instead of learnable filter banks. It was proved in [3] that scattering transforms are Lipschitz stable with respect to smooth deformations of space (i.e., they are stable with respect to the gradient of the diffeomorphism of the domain). This stability property is shared by linear wavelet banks, only if their frequency responses are flat at high frequencies [4]. One can restrict attention to this class of filters but only at the cost of losing the ability to discriminate high frequency features. The nonlinear operation in the scattering transform acts as a frequency mixer that brings part of the high frequency energy towards low frequencies where it can be discriminated with stable filters. Thus, scattering transforms can be, both, stable and discriminative, but wavelet banks cannot be simultaneously stable and discriminative. The similarities between scattering transforms and CNNs (both computed as convolutions followed by nonlinearities) suggest that this conclusion can be extrapolated to CNNs, in the sense that we can argue that they improve over linear filters because they are simultaneously stable and discriminative. Recent developments in computer vision similarly link the frequency content of images (Euclidean data) with their stability [5], [6].

Parallel to the development of CNNs, the field of graph signal processing (GSP) has emerged as a generalization of Euclidean signal processing to signals whose components are related by arbitrary pairwise relationships described by an underlying graph support [7], [8]. Central to GSP is the generalization of linear convolutional filters as polynomials of some matrix representation of the graph [9]. Having a valid convolution operation the notion of a graph neural network (GNN) emerges naturally as a cascade of layers, where each layer is made up of a graph convolution filter bank composed with a pointwise nonlinearity [10]–[13]. GNNs have, predictably, proved useful in a variety of problems where they, surprisingly, outperform linear graph filters [14]–[18].

The main contribution of this article is to show that the advantage of GNNs relative to linear graph filters is their stability to graph deformations (Theorem 4). Our analysis utilizes graph Fourier transforms to provide a representation of the filter on the spectrum of the matrix representation of the graph. This representation shows that graph filters cannot be stable if they are designed to isolate features associated with large eigenvalues

Manuscript received September 17, 2019; revised April 22, 2020 and July 8, 2020; accepted September 17, 2020. Date of publication September 25, 2020; date of current version October 9, 2020. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Vincent Gripon. Fernando Gama and Alejandro Ribeiro are supported by NSF CCF 1717120, ARO W911NF1710438, ARL DCIST CRA W911NF-17-2-0181, ISTC-WAS and Intel DevCloud. Joan Bruna is partially supported by the Alfred P. Sloan Foundation, NSF RI-1816753, NSF CAREER CIF 1845360, and Samsung Electronics. *(Corresponding author: Fernando Gama.)*

Fernando Gama is with the Electrical Engineering and Computer Sciences Department, University of California, Berkeley, CA 94709, USA (e-mail: fgama@berkeley.edu).

Joan Bruna is with the Courant Institute of Mathematical Sciences and Center for Data Science, New York University, New York, NY 10003 USA (e-mail: bruna@cims.nyu.edu).

Alejandro Ribeiro is with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: aribeiro@seas.upenn.edu).

Digital Object Identifier 10.1109/TSP.2020.3026980

of the graph (Theorem 2). This is the equivalent of Euclidean convolutional filters being unable to be stable if they discriminate high frequencies. The graph filter banks that are used by GNNs cannot discriminate features associated with large eigenvalues either. But pointwise nonlinearities perform frequency mixing that brings part of the energy associated with large eigenvalues towards low eigenvalues where it can be discriminated with stable graph filters. Thus, GNNs can be, both, stable and discriminative, but linear graph filters cannot be simultaneously stable and discriminative. This is analogous to the reasons that explain why scattering transforms (and, by extension, CNNs) have an advantage with respect to linear Euclidean filters [3]–[6].

The analysis of stability properties for the case of non-trainable GNNs built with graph wavelet filter banks has been carried out by [19], [20], in analogy to [3], [4]. More specifically, [19] studies the stability of these GNNs to permutations, as well as to perturbations on the eigenvalues and eigenvectors of the underlying graph support. Alternatively, [20] considers the specific case of using diffusion wavelets and proves permutation invariance as well as stability to perturbations measured by the diffusion distance [21]. Both of these works consider an absolute perturbation model, where changes in the underlying graph support do not take into account the particularities of the topology. This leads to results that either depend on the size of the graph (i.e. larger graphs admit smaller edge weight changes) [19] or on the spectral gap [20], tying the applicability of the results to the specific graph under consideration. Stability of arbitrary filter banks has been studied in [22] by leveraging a bound in [20, eq. (23)]. This result also depends on the spectral gap. Permutation equivariance has been studied in [23]–[25]. In particular, [23] considers the question of graph isomorphisms by means of the Weisfeiler-Lehman test, [24] characterizes the space of invariant and equivariant linear layers, and [25] extends the result in [24] to graphs of varying size.

The underlying support can change due to targeted attacks on the nodes and edges of the graph. The robustness of GNNs to these malicious, adversarial attacks is being studied. In [26], the focus is on designing adversarial attacks such that the label of a target node is misclassified by carefully changing the edges and signal values of other nodes. It considers binary adjacency matrices and binary graph signals, and assumes that the semi-supervised problem is solved by means of a GCN [12] with a single-hidden layer. The work in [27] also focuses on designing adversarial attacks, but uses reinforcement learning and focuses on the problems of both node and graph classification. In the case when the attacks are crafted to focus on a subset of edges in a semi-supervised learning problem, where labels are inferred using GNNs with IIR filters on binary adjacency matrices, [28] provides robustness certificates for which nodes will not change the learned label under these attacks, and also proposes robust training of the model by adding a penalty to the cross-entropy loss function. Robustness certificates and robust training have also been developed for adversarial attacks on the binary signal values of a semi-supervised learning problem, and where labels are obtained by means of a GCN [29]. In this article, however, we focus on changes that can occur due to topology inference errors or due to time-varying scenarios, instead of crafted attacks.

We begin the article in Section II by showing that linear graph filters are equivariant to permutations (Prop. 1). Then, we discuss the model of absolute perturbations modulo permutation (analogous to that in [19], [20]) and show that a linear filter whose frequency response is Lipschitz continuous is stable (Theorem 1), with a constant that depends on the Lipschitz condition of the filters as well as the intrinsic topological characteristics of the graph and its perturbation. Next, we show that, under the relative perturbation model, integral Lipschitz graph filters are stable (Theorem 2), and determine a family of perturbations under which the stability can be entirely controlled by the integral Lipschitz constant of the filters, for any graph (Theorem 3). In Section III we show how the stability results for graph filters carry over to GNN architectures (Theorem 4). Section IV offers an insightful discussion on the results, showing that a GNN built on Lipschitz filters under an absolute perturbation model exhibits a trade-off between stability and discriminability, while another one using integral Lipschitz filters under a relative perturbation model can be made simultaneously stable and discriminative. Finally, we numerically illustrate stability in a problem of movie recommendation (Section V), and conclude (Section VI).

II. Stability Properties of Graph Filters

We work with graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ described by a set of N nodes $\mathcal{V}$, a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and a weight function $\mathcal{W} : \mathcal{E} \to \mathbb{R}$. We can associate to $\mathcal{G}$ a matrix representation $\mathbf{S} \in \mathbb{R}^{N \times N}$ that respects the sparsity of the graph, namely, $s_{ij} = [\mathbf{S}]_{ij} = 0$ whenever $i \neq j$ or $(j, i) \notin \mathcal{E}$. This is a condition that is verified by, e.g., adjacency matrices, Laplacians, random walk Laplacians, and their normalized counterparts. We generically call $\mathbf{S}$ a graph shift operator (GSO) [7]. We assume the shift operator is symmetric with eigenvector basis $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_N]$ and eigenvalue matrix $\mathbf{\Lambda} = \text{diag}([\lambda_1, \ldots, \lambda_N])$ so that we can write

$$\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathsf{H}}. \tag{1}$$

It is assumed that eigenvalues are ordered from smallest to largest so that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$.

The graph acts as a support for the data vector $\mathbf{x} \in \mathbb{R}^N$ which we henceforth say to be a graph signal $\mathbf{x} = [x_1, \cdots, x_N]^{\mathsf{T}}$ that assigns the value x_i to node i. The shift operator $\mathbf{S}$ defines a linear map $\mathbf{y} = \mathbf{S}\mathbf{x}$ between graph signals that represents the local exchange of information between a node and its neighbors. Repeated application of $\mathbf{S}$ accesses information from nodes located farther away since the product $\mathbf{S}^k\mathbf{x} = \mathbf{S}(\mathbf{S}^{k-1}\mathbf{x})$ represents the aggregation at node i of information located in nodes of its k-hop neighborhood. Aggregating information from k-hop neighbors is analogous to applying k time shifts to a time signal. This is the motivation for defining graph convolutional filters as polynomials on the shift operator that, for a set of coefficients $\mathbf{h} = \{h_k\}_{k=0}^{\infty}$ process input graph signals $\mathbf{x}$ to produce output graph signals,

$$\mathbf{z} \;=\; \sum_{k=0}^{\infty} h_k \mathbf{S}^k \mathbf{x} \;:=\; \mathbf{H}(\mathbf{S})\mathbf{x}. \tag{2}$$

The matrix $\mathbf{H}(\mathbf{S}) := \sum_{k=0}^{\infty} h_k \mathbf{S}^k$ in (2) is said to be a finite impulse response (FIR) graph filter or a graph convolutional filter, and the coefficients h_k are thus called *filter taps* or *filter weights* [9]. A set of coefficients $\mathbf{h}$, defines a filter $\mathbf{H}(\mathbf{S})$ for any given graph $\mathbf{S}$. In particular, if we are given another graph $\hat{\mathbf{S}}$ we can construct the filter $\mathbf{H}(\hat{\mathbf{S}}) := \sum_{k=0}^{\infty} h_k \hat{\mathbf{S}}^k$, with the same filter taps $\mathbf{h}$, whose application to graph signals $\hat{\mathbf{x}}$ produces graph signals

$$\hat{\mathbf{z}} = \sum_{k=0}^{\infty} h_k \hat{\mathbf{S}}^k \hat{\mathbf{x}} := \mathbf{H}(\hat{\mathbf{S}})\hat{\mathbf{x}}. \tag{3}$$

We want to characterize the difference between filters $\mathbf{H}(\mathbf{S})$ and $\mathbf{H}(\hat{\mathbf{S}})$ in terms of the differences between shift operators $\mathbf{S}$ and $\hat{\mathbf{S}}$. We study the effect of permutations in Section II-A and the effect of perturbations in Sections II-B and II-C.

A. Permutation Equivariance of Graph Filters

For a given dimension N we define permutation matrices $\mathbf{P}$ as those that belong to the set

$$\mathcal{P} = \left\{ \mathbf{P} \in \{0,1\}^{N \times N} : \mathbf{P}\mathbf{1} = \mathbf{1}, \mathbf{P}^{\mathsf{T}}\mathbf{1} = \mathbf{1} \right\}. \tag{4}$$

As per (4), a permutation matrix $\mathbf{P}$ is one in which the product $\hat{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{x}$ is a reordering of the entries of the vector $\mathbf{x}$. Likewise, for a given shift operator $\mathbf{S}$, the shift operator $\hat{\mathbf{S}} = \mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P}$ is a reordering of the rows and columns of $\mathbf{S}$. Thus, the graph $\hat{\mathbf{S}}$ is just a relabeling of the nodes of $\mathbf{S}$. Taken together, the graph $\hat{\mathbf{S}} = \mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P}$ and the signal $\hat{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{x}$ represent a consistent relabeling of the graph $\mathbf{S}$ and the graph signal $\mathbf{x}$. One would expect that the application of the filter on the relabeled graph to the relabeled signal produces an output that corresponds to the relabeled output prior to permutation of the graph. The following proposition asserts that this is true.

Proposition 1 (Permutation equivariance): Consider graph shifts $\mathbf{S}$ and $\hat{\mathbf{S}} = \mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P}$ for some permutation matrix $\mathbf{P} \in \mathcal{P}$ [cf. (4)]. Given a set of coefficients $\mathbf{h}$, the filters $\mathbf{H}(\mathbf{S})$ and $\mathbf{H}(\hat{\mathbf{S}})$ in (2) and (3) are such that for any pair of corresponding graph signals $\mathbf{x}$ and $\hat{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{x}$ the graph filter outputs $\mathbf{z} := \mathbf{H}(\mathbf{S})\mathbf{x}$ and $\hat{\mathbf{z}} := \mathbf{H}(\hat{\mathbf{S}})\hat{\mathbf{x}}$ satisfy

$$\hat{\mathbf{z}} := \mathbf{H}(\hat{\mathbf{S}})\hat{\mathbf{x}} := \mathbf{H}(\hat{\mathbf{S}})(\mathbf{P}^{\mathsf{T}}\mathbf{x}) = \mathbf{P}^{\mathsf{T}}(\mathbf{H}(\mathbf{S})\mathbf{x}) := \mathbf{P}^{\mathsf{T}}\mathbf{z}. \tag{5}$$

Proof: See Appendix A. ∎

Proposition 1 states the permutation equivariance of graph filters. Namely, a permutation of the input – from $\mathbf{x}$ to $\hat{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{x}$ – along with a permutation of the shift operator – from $\mathbf{S}$ to $\hat{\mathbf{S}} = \mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P}$ – results in a permutation of the output – from $\mathbf{z}$ to $\hat{\mathbf{z}} = \mathbf{P}^{\mathsf{T}}\mathbf{z}$. Notice that if we are given vectors $\mathbf{x}$ and $\hat{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{x}$ and we know the permutation matrix $\mathbf{P}$ it is elementary to design linear operators that are permutation equivariant by simply applying the permutation to the linear operator. The permutation equivariance in (5) is more subtle in that it holds without having access to the permutation $\mathbf{P}$.

Permutation equivariance of graph filters implies their usefulness in applications where graph relabeling is inconsequential. This motivates consideration of the space of linear operators modulo permutation along with operator distances between these equivalence classes which we introduce next.

Definition 1 (Linear operator distance modulo permutation): Given linear operators $\mathbf{A}$ and $\hat{\mathbf{A}}$ we define the operator distance modulo permutation as

$$\left\| \mathbf{A} - \hat{\mathbf{A}} \right\|_{\mathcal{P}} = \min_{\mathbf{P} \in \mathcal{P}} \max_{\mathbf{x}: \|\mathbf{x}\|=1} \left\| \mathbf{P}^{\mathsf{T}}(\mathbf{A}\mathbf{x}) - \hat{\mathbf{A}}(\mathbf{P}^{\mathsf{T}}\mathbf{x}) \right\|. \tag{6}$$

The distance in (6) compares the effect of applying $\mathbf{A}$ and $\hat{\mathbf{A}}$ to a vector $\mathbf{x}$ with a permutation $\mathbf{P}$ applied before or after application of the linear operators. It measures this difference at the unit norm vector for which it is largest and at the permutation matrix that makes it smallest. We can further denote as $\mathbf{P}_0 \in \mathcal{P}$ a matrix that attains the minimum in (6) and define the *absolute perturbation modulo permutation* as

$$\mathbf{E} = \mathbf{A} - \mathbf{P}_0^{\mathsf{T}} \hat{\mathbf{A}} \mathbf{P}_0. \tag{7}$$

The perturbation matrix $\mathbf{E}$ is the difference between operators $\mathbf{A}$ and $\mathbf{P}_0^{\mathsf{T}}\hat{\mathbf{A}}\mathbf{P}_0$ for the permutation matrix $\mathbf{P}_0$ that achieves the minimum norm difference in (6). If there is more than one matrix that achieves the minimum, an arbitrary choice is acceptable for the results we will derive to hold. Further notice that it follows from (6) and (7) that the operator distance between $\mathbf{A}$ and $\hat{\mathbf{A}}$ is simply the operator norm of $\mathbf{E}$,

$$\left\| \mathbf{A} - \hat{\mathbf{A}} \right\|_{\mathcal{P}} = \|\mathbf{E}\|. \tag{8}$$

It is ready to see that Def. 1 is a proper distance in the space of linear operators modulo permutation. In particular, it holds that $\left\| \mathbf{A} - \hat{\mathbf{A}} \right\|_{\mathcal{P}} = 0$ if and only there exists a permutation matrix for which $\hat{\mathbf{A}} = \mathbf{P}^{\mathsf{T}}\mathbf{A}\mathbf{P}$. From this latter fact it follows that we can rewrite Prop. 1 to say that the distance modulo permutation between graph filters $\mathbf{H}(\mathbf{S})$ and $\mathbf{H}(\hat{\mathbf{S}})$ is null if the distance modulo permutation between the shift operators $\mathbf{S}$ and $\hat{\mathbf{S}}$ is null. We formally state and prove this fact next.

Corollary 1 (Permutation equivariance): For shift operators $\mathbf{S}$ and $\hat{\mathbf{S}}$ whose distance modulo permutation is $\|\mathbf{S} - \hat{\mathbf{S}}\|_{\mathcal{P}} = 0$, the distance modulo permutation between graph filters $\mathbf{H}(\mathbf{S})$ and $\mathbf{H}(\hat{\mathbf{S}})$ satisfies

$$\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}} = 0. \tag{9}$$

Proof: If $\|\mathbf{S} - \hat{\mathbf{S}}\|_{\mathcal{P}} = 0$ there exists permutation $\mathbf{P}$ such that $\hat{\mathbf{S}} = \mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P}$ and Proposition 1 holds. Thus, for this same permutation we have $\mathbf{P}^{\mathsf{T}}(\mathbf{H}(\mathbf{S})\mathbf{x}) = \mathbf{H}(\hat{\mathbf{S}})(\mathbf{P}^{\mathsf{T}}\mathbf{x})$ for any vector $\mathbf{x}$. The result in (9) then follows from Def. 1. ∎

Corollary 1 says that if two graphs are the same, the graph filters are also the same, modulo permutation. In the next section we address the question of what happens to the filter difference $\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}}$ when the difference $\|\mathbf{S} - \hat{\mathbf{S}}\|_{\mathcal{P}}$ is small but not null. Namely, we investigate the similarity of $\mathbf{H}(\mathbf{S})$ and $\mathbf{H}(\hat{\mathbf{S}})$ when the shift operators $\mathbf{S}$ and $\hat{\mathbf{S}}$ are close to being permuted versions of each other.

B. Effect of Absolute Graph Perturbations on Graph Filters

To understand the stability of graph filters it is instructive to consider the form of (2) in the graph frequency domain. This entails using the eigenvector basis $\mathbf{V}$ in (1) to define the *Graph Fourier Transform* (GFT) of the graph signal $\mathbf{x}$ as the projection $\tilde{\mathbf{x}} = \mathbf{V}^{\mathsf{H}}\mathbf{x}$ [8]. Substituting the GFT definition in the definition of the graph convolution in (2) and using the fact that $\mathbf{S}^k =$

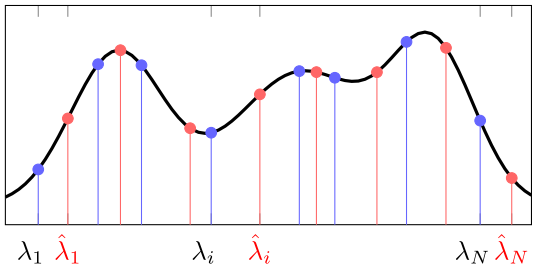


Fig. 1. Frequency response of a graph filter. The function $h(\lambda)$ is shown as a black, solid line, and is independent of the graph. When evaluated on a given graph shift operator, specific values of h are instantiated on the eigenvalues of the given GSO. For example, when given a GSO $\mathbf{S}$ with eigenvalues $\{\lambda_i\}$, the graph filter frequency response will be instantiated on $h(\lambda_i)$ (in blue); but, if we are given a different GSO $\hat{\mathbf{S}}$ with other eigenvalues $\{\hat{\lambda}_i\}$, then the filter frequency response will be given by $h(\hat{\lambda}_i)$ (in red).

$\mathbf{V}\mathbf{\Lambda}^k\mathbf{V}^{\mathsf{H}}$ we can write

$$\mathbf{V}^{\mathsf{H}}\mathbf{z} \;=\; \tilde{\mathbf{z}} \;=\; \sum_{k=0}^{\infty} h_k \mathbf{\Lambda}^k \left(\mathbf{V}^{\mathsf{H}}\mathbf{x}\right) \;=\; \mathbf{H}(\mathbf{\Lambda})\hat{\mathbf{x}}. \tag{10}$$

The interesting conclusion that follows from (10) is that graph filters are pointwise operators in the graph frequency domain because $\mathbf{H}(\mathbf{\Lambda})$ is a diagonal matrix. This motivates the definition of the *graph frequency response* of the filter as

$$h(\lambda) \;=\; \sum_{k=0}^{\infty} h_k \lambda^k. \tag{11}$$

Comparing (10) with (11) we conclude that the ith component $\tilde{x}_i$ of the input signal GFT $\tilde{\mathbf{x}}$ and the ith component $\tilde{z}_i$ of the output signal GFT $\tilde{\mathbf{z}}$ are related through the expression

$$\tilde{z}_i = h(\lambda_i)\tilde{x}_i. \tag{12}$$

The remarkable observation to be made at this point is that the frequency response of a filter is completely characterized by the filter coefficients $\mathbf{h}$ [cf. (11)]. The effect of a specific graph is to determine the components of the GFT $\tilde{\mathbf{x}} = \mathbf{V}^{\mathsf{H}}\mathbf{x}$ – through its eigenvectors $\mathbf{V}$ – and which values of the frequency response are instantiated [cf. (12)] – through its eigenvalues $\mathbf{\Lambda}$. Fig. 1 shows an illustration of this latter fact. We have a filter with frequency response $h(\lambda)$ represented as a continuous function. For a graph with eigenvalues λ_i only the values at frequencies $h(\lambda_i)$ affect the response of the filter. For a different graph with eigenvalues $\hat{\lambda}_i$ the values $h(\hat{\lambda}_i)$ are the ones that determine the effect of the filter in the given graph.

Since graph perturbations alter the spectrum of a graph it seems apparent that the variability of the frequency response $h(\lambda)$ has a direct effect on a filter's stability to perturbations. To proceed with a formal characterization we introduce the notion of Lipschitz filters in the following definition.

Definition 2 (Lipschitz Filter): Given a filter $\mathbf{h} = \{h_k\}_{k=0}^{\infty}$ its frequency response $h(\lambda)$ is given by (11) and satisfies $|h(\lambda)| \leq 1$. We say the filter is Lipschitz if there exists a constant $C > 0$ such that for all λ_1 and λ_2 the frequency response is such that

$$\left|h(\lambda_2) - h(\lambda_1)\right| \leq C\left|\lambda_2 - \lambda_1\right|. \tag{13}$$

As its name suggests, a filter is Lipschitz if its frequency response is Lipschitz. This means a Lipschitz filter is one whose frequency response does not change faster than linear. For filters that are Lipschitz, the following stability result relative to perturbations that are close to permutations holds.

Theorem 1: Let $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathsf{H}}$ and $\hat{\mathbf{S}}$ be graph shift operators. Let $\mathbf{E} = \mathbf{U}\mathbf{M}\mathbf{U}^{\mathsf{H}}$ be the absolute perturbation modulo permutation between $\mathbf{S}$ and $\hat{\mathbf{S}}$ [cf. (7)] and assume their operator distance modulo permutation (cf. Def. 1) satisfies

$$\|\mathbf{S} - \hat{\mathbf{S}}\|_{\mathcal{P}} = \|\mathbf{E}\| \leq \varepsilon. \tag{14}$$

For a Lipschitz filter (cf. Def. 2) with Lipschitz constant C the operator distance modulo permutation between filters $\mathbf{H}(\mathbf{S})$ and $\mathbf{H}(\hat{\mathbf{S}})$ satisfies

$$\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}} \;\leq\; C\left(1 + \delta\sqrt{N}\right)\varepsilon + \mathcal{O}(\varepsilon^2) \tag{15}$$

with $\delta := (\|\mathbf{U} - \mathbf{V}\|_2 + 1)^2 - 1$ standing for the eigenvector misalignment between shift operator $\mathbf{S}$ and error matrix $\mathbf{E}$.

Proof: See Appendix B. ∎

To a first order approximation, Theorem 1 shows that graph filters are Lipschitz stable with respect to absolute graph perturbations [cf. (7)–(8)]. The stability constant is given by $C(1 + \delta\sqrt{N})$ which implies that (i) the bound holds uniformly for all graphs with N nodes, (ii) it is affected by a term, the Lipschitz constant C, that is controllable through filter design, and (iii) it is further affected by a term, the eigenvector misalignment $(1 + \delta\sqrt{N})$, that depends on the structure of the perturbations that are expected in a particular problem but that cannot be affected by judicious filter choice. We note that the Lipschitz stability established by Theorem 1 is with respect to the changes in the underlying graph support $\mathbf{S}$, and not with respect to changes in the input $\mathbf{x}$.

Although Theorem 1 shows filter stability with respect to graph perturbations, the stability claim may be misleading given that the perturbation's norm is not tied to the norm of the graph shift. To do so we replace (14) with the hypothesis $\|\mathbf{S} - \hat{\mathbf{S}}\|_{\mathcal{P}} = \|\mathbf{E}\| \leq \varepsilon\|\mathbf{S}\|$, under which (15) becomes

$$\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}} \;\leq\; C\left(1 + \delta\sqrt{N}\right)\|\mathbf{S}\|\varepsilon + \mathcal{O}(\varepsilon^2). \tag{16}$$

The bound in (16) is the result of a *relative* perturbation model. It is still a stability result but, in contrast to (15), not one that is uniform for all graphs with a given number of nodes. Making $\|\mathbf{S}\|$ arbitrarily large, makes the constant $(1 + \delta\sqrt{N})\|\mathbf{S}\|$ arbitrarily large. One could think that it is reasonable to allow for larger filter perturbations when graphs have larger norm but this is not true. Filter perturbations determine feature perturbations whose magnitude need not be related to the graph's norm. On closer inspection the problem with making $\|\mathbf{E}\| \leq \varepsilon\|\mathbf{S}\|$ is that the norms of $\mathbf{E}$ and $\mathbf{S}$ are global properties of the error and the graph. In particular, this may imply that parts of the graph with small weights have large relative modifications because some other parts of the graph have large weights. This observation prompts the relative perturbation model which effectively ties the local properties of the shift operator and error matrices, as discussed next.

C. Effect of Relative Graph Perturbations on Graph Filters

We tie the perturbation of edges to their magnitudes by considering a relative perturbation model such that the relationship between a graph $\mathbf{S}$ and its perturbed version $\hat{\mathbf{S}}$ is given by the error matrices $\mathbf{E}$ in the following set.

Definition 3 (Relative Perturbation Modulo Permutation): Given shift operators $\mathbf{S}$ and $\hat{\mathbf{S}}$ we define the set of relative perturbation matrices modulo permutation as

$$\mathcal{E}(\mathbf{S},\hat{\mathbf{S}}) = \left\{\mathbf{E} : \mathbf{P}^{\mathsf{T}}\hat{\mathbf{S}}\mathbf{P} = \mathbf{S} + (\mathbf{E}\mathbf{S} + \mathbf{S}\mathbf{E}), \mathbf{P} \in \mathcal{P}\right\}. \quad (17)$$

The set of relative perturbation matrices modulo permutation considers all the matrices $\mathbf{E}$ that allow us to write different permutations of $\hat{\mathbf{S}}$ through relative perturbations of $\mathbf{S}$ given by the model in (17). The norm $\|\mathbf{E}\|$ of a given error matrix $\mathbf{E}$ associated to a given permutation $\mathbf{P}$ is a measure of relative dissimilarity between $\mathbf{P}^{\mathsf{T}}\hat{\mathbf{S}}\mathbf{P}$ and $\mathbf{S}$. To measure dissimilarity between $\mathbf{S}$ and $\hat{\mathbf{S}}$ modulo permutation we evaluate the error norm at the permutation that affords the smallest error norm

$$\begin{aligned} d(\mathbf{S},\hat{\mathbf{S}}) = \min_{\mathbf{P}\in\mathcal{P}} \quad & \|\mathbf{E}\| \\ \text{s.t.} \quad & \mathbf{P}^{\mathsf{T}}\hat{\mathbf{S}}\mathbf{P} = \mathbf{S} + (\mathbf{E}\mathbf{S} + \mathbf{S}\mathbf{E}). \end{aligned} \quad (18)$$

The dissimilarity $d(\mathbf{S},\hat{\mathbf{S}})$ measures how close $\mathbf{S}$ and $\hat{\mathbf{S}}$ are to being permutations of each other, as determined by the multiplicative factor $\mathbf{E}$. Such a model ties changes in the edge weights of the graph to its local structure. To see this, note that the difference between the edge weight s_{ij} of the original graph $\mathbf{S}$ and the corresponding edge $[\mathbf{P}_0^{\mathsf{T}}\hat{\mathbf{S}}\mathbf{P}_0]_{ij}$ of the perturbed graph $\hat{\mathbf{S}}$ is given by the corresponding entry $[\mathbf{E}\mathbf{S} + \mathbf{S}\mathbf{E}]_{ij}$ of the perturbation factor $\mathbf{E}\mathbf{S} + \mathbf{S}\mathbf{E}$. It is ready to see that this quantity is proportional to the sum of the degrees of nodes i and j scaled by the entries of $\mathbf{E}$. As the norm $\|\mathbf{E}\|$ grows, the entries of the graphs $\mathbf{S}$ and $\mathbf{P}_0^{\mathsf{T}}\hat{\mathbf{S}}\mathbf{P}_0$ become more dissimilar. But parts of the graph that are characterized by weaker connectivity change by amounts that are proportionally smaller to the changes that are observed in parts of the graph characterized by stronger links. This is in contrast to absolute perturbations where edge weights change by the same amount irrespective of the local topology of the graph.

To study the effect of relative perturbations we can rely on the consideration of Lipschitz filters (cf. Def. 2) but more illuminating results are possible with the use of integral Lipschitz filters, which satisfy a condition on the rate of change of their frequency responses that we introduce next.

Definition 4 (Integral Lipschitz Filter): Given a filter $\mathbf{h} = \{h_k\}_{k=0}^{\infty}$ its frequency response $h(\lambda)$ is given by (11) and satisfies $|h(\lambda)| \leq 1$. We say the filter is integral Lipschitz if there exists a constant $C > 0$ such that for all λ_1 and λ_2,

$$|h(\lambda_2) - h(\lambda_1)| \leq C \frac{|\lambda_2 - \lambda_1|}{|\lambda_1 + \lambda_2|/2}. \quad (19)$$

The condition in (19) can be read as requiring the filter's frequency response to be Lipschitz in any interval (λ_1, λ_2) with a Lipschitz constant that is inversely proportional to the interval's midpoint $(|\lambda_1 + \lambda_2|)/2$. To see this better, observe that (19) restricts the frequency response's derivative to satisfy,

$$\left|\lambda h'(\lambda)\right| \leq C. \quad (20)$$

Thus, filters that are integral Lipschitz must have frequency responses that have to be flat for large λ but can vary very rapidly around $\lambda = 0$. This is, not coincidentally, a condition reminiscent of the scale invariance of wavelet transforms [30, Ch. 7]. The condition $|h(\lambda)| \leq 1$ is not necessary but it eases interpretations by preventing the filter from amplifying energy.

For relative perturbation models and integral Lipschitz filters the following stability result holds.

Theorem 2: Let $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathsf{H}}$ and $\hat{\mathbf{S}}$ be graph shift operators. Let $\mathbf{E} = \mathbf{U}\mathbf{M}\mathbf{U}^{\mathsf{H}} \in \mathcal{E}(\mathbf{S},\hat{\mathbf{S}})$ be a relative perturbation matrix (cf. Def. 3) whose norm is such that [cf. (18)]

$$d(\mathbf{S},\hat{\mathbf{S}}) \leq \|\mathbf{E}\| \leq \varepsilon. \quad (21)$$

For an integral Lipschitz filter (cf. Def. 4) with integral Lipschitz constant C the operator distance modulo permutation between filters $\mathbf{H}(\mathbf{S})$ and $\mathbf{H}(\hat{\mathbf{S}})$ satisfies

$$\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}} \leq 2\, C \left(1 + \delta\sqrt{N}\right)\varepsilon + \mathcal{O}(\varepsilon^2) \quad (22)$$

with $\delta := (\|\mathbf{U} - \mathbf{V}\|_2 + 1)^2 - 1$ standing for the eigenvector misalignment between shift operator $\mathbf{S}$ and error matrix $\mathbf{E}$

Proof: See Appendix C. ■

Theorem 2 establishes stability with respect to relative perturbations of the form introduced in Def. 3. If a matrix $\mathbf{E}$ exists that makes $\mathbf{S}$ and $\hat{\mathbf{S}}$ close to permutations of each other in terms of this relative perturbation, the filters are stable with respect to the norm of the perturbation with stability constant $2\,C(1 + \delta\sqrt{N})$. The constant has the same shape as the one in Theorem 1, and while this is a coincidence, similar observations hold. Namely, the bound is uniform for all graphs with the same number of nodes, the stability is affected by the integral Lipschitz constant C which depends on the filter, and it is also affected by the eigenvector misalignment $(1 + \delta\sqrt{N})$, which depends on the structure of the perturbation. The important difference between Thms. 1 and 2 is that the meaning of C is different since the class of filters that are admissible for stability with respect to relative perturbations is that of integral Lipschitz filters – whereas Lipschitz filters are required for stability with respect to absolute perturbations.

Before elaborating on the implications of allowing for integral Lipschitz filters we consider a variation of Theorem 2 in which we impose a structural constraint on the perturbation.

Theorem 3: With the same hypotheses and definitions of Theorem 2 assume that there exists a matrix $\mathbf{E} \in \mathcal{E}(\mathbf{S},\hat{\mathbf{S}})$ that satisfies (21) and, furthermore, is such that

$$\min\left[\left\|\frac{\mathbf{E}}{\|\mathbf{E}\|} - \mathbf{I}\right\|, \left\|\frac{\mathbf{E}}{\|\mathbf{E}\|} + \mathbf{I}\right\|\right] \leq \varepsilon. \quad (23)$$

Then, the operator distance modulo permutation between filters $\mathbf{H}(\mathbf{S})$ and $\mathbf{H}(\hat{\mathbf{S}})$ satisfies

$$\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}} \leq 2\, C\varepsilon + \mathcal{O}(\varepsilon^2). \quad (24)$$

Proof: See Appendix C. ■

The structural constraint in (23) requires the error matrix $\mathbf{E}$ to be a scaled identity to within a first order approximation. With this restriction on the set of admissible perturbations we can bound the eigenvector misalignment between $\mathbf{S}$ and $\mathbf{E}$ and remove the dependency that the bound in Theorem 2 has on the number of nodes N. The bound in (24) holds uniformly for all graphs independently of their number of nodes.

The integral Lipschitz filters in Thms. 2 and 3 are of interest because they can be made finely discriminative at low-eigenvalue frequencies without affecting stability. Indeed, to control stability in (22) we need to limit the value of the integral Lipschitz constant C. This requires filters that change more slowly. In particular, for large λ the filters must be constant and cannot discriminate nearby spectral features. But at values of $\lambda \approx 0$ the filters can change rapidly and can therefore be designed to discriminate (arbitrarily) close spectral features. To the extent that relative perturbations are admissible – which, as already discussed, are arguably more sensible than absolute ones – Theorem 2 shows two fundamental properties of linear graph filters: (i) They *cannot* be stable and discriminative of spectral features associated with large λ. (ii) They *can* be stable and discriminative of spectral features associated with $\lambda \approx 0$.

Thus, if we are interested in discriminating features associated with $\lambda \approx 0$, linear graph filters are sufficient. However, if we are interested in discriminating features associated with large λ, linear graph filters will fail because of their sensitivity to graph perturbations. We will see in the following section that this is an issue we can resolve with the introduction of pointwise nonlinearities to produce graph neural networks.

Remark 1 (Integral Lipschitz filters): We note that filter banks abiding to the integral Lipschitz condition exist in the literature; see, for example, graph wavelets [31]–[33]. However, filters in a GNN are trained from data and it is therefore not necessarily guaranteed that they will satisfy this condition. To address this from a practical standpoint, in Section V we add the integral Lipschitz condition as a penalty during training (20). This results in filters with controllable constant C that illustrate different degrees of stability. We further remark that, regardless of enforcing integral Lipschitz conditions on the filters learned in a GNN, the insights stemming from the discussion ensuing in Section IV still hold. Namely, that GNNs are simultaneously stable and discriminative, a feat that cannot be achieved by linear filter banks by themselves.

Remark 2 (Eigenvector misalignment): The bound (22) in Theorem 2 depends on the eigenvector misalignment constant δ. This, in turn, depends on the specific relative perturbation $\mathbf{E} \in \mathcal{E}$ [cf. (17)] and computing it requires an eigendecomposition. However, there are two important observations to be made. First, that while Theorem 2 holds for all values of $\mathbf{E}$, there are specific families of perturbations for which the eigenvector misalignment constant can be known (as is the case with Euclidean perturbations). Second, that regardless of the specific $\mathbf{E}$, it always holds that $\delta \leq 8$, following from the fact that $\|\mathbf{U}\| \leq 1$ and $\|\mathbf{V}\| \leq 1$ because they are eigenvector matrices. However, this accentuates the dependence of the Lipschitz constant with the size of the graph N.

III. Stability Properties of Graph Neural Networks

Graph neural networks are a cascade of layers, each of which applies a bank of graph filters, followed by a pointwise nonlinearity [10]–[13]. To increase the representation power of GNNs, we consider that at layer ℓ there are F_ℓ graph signals $\mathbf{x}_\ell^f \in \mathbb{R}^N$, $f = 1, \ldots, F_\ell$ instead of a single one, as was the case in the previous section. Each graph signal $\mathbf{x}_\ell^f$ is called a *feature*. The input to layer ℓ is then the $F_{\ell-1}$ signals $\mathbf{x}_{\ell-1}^g$ that were the output of layer $\ell - 1$, $g = 1, \ldots, F_{\ell-1}$. To compute the F_ℓ features $\mathbf{x}_\ell^f$ we expect at the output of layer ℓ, we first process the $F_{\ell-1}$ input signals with a bank of $F_{\ell-1}F_\ell$ graph filters denoted by $\mathbf{H}_\ell^{fg}(\mathbf{S})$ [cf. (2)], defined by coefficients $\mathbf{h}_\ell^{fg}$

$$\mathbf{z}_\ell^{fg} = \sum_{k=0}^{\infty} h_{\ell k}^{fg} \mathbf{S}^k \mathbf{x}_{\ell-1}^g = \mathbf{H}_\ell^{fg}(\mathbf{S}) \mathbf{x}_{\ell-1}^g \tag{25}$$

where we obtain the intermediate features $\mathbf{z}_\ell^{fg}$ for $f = 1, \ldots, F_\ell$ and $g = 1, \ldots, F_{\ell-1}$. All of these intermediate features $\mathbf{z}_\ell^{fg}$ for a given index f are summed together to linearly yield F_ℓ features, and passed through a pointwise nonlinear function $\sigma : \mathbb{R} \to \mathbb{R}$ to produce the output feature

$$\mathbf{x}_\ell^f = \sigma \left[\sum_g \mathbf{z}_\ell^{fg} \right] \tag{26}$$

for $f = 1, \ldots, F_\ell$. We note that, in an abuse of notation, the application of σ to the vector $\sum_g \mathbf{z}_\ell^{fg}$ implies the entrywise application of the nonlinearity, i.e. the same σ is applied independently to every feature at every node. The input to the GNN is the 0th layer signal $\mathbf{x} = \mathbf{x}_0^1$ and the output of the GNN is the Lth layer feature $\mathbf{x}_L^1$.

We note that (25)–(26) can be compactly written as

$$\mathbf{X}_\ell = \sigma \left[\sum_{k=0}^{\infty} \mathbf{S}^k \mathbf{X}_{\ell-1} \mathbf{H}_{\ell k} \right] \tag{27}$$

for $\mathbf{X}_\ell \in \mathbb{R}^{N \times F_\ell}$ the matrix whose columns are the graph signal features $\mathbf{x}_\ell^f$ and where $\mathbf{H}_{\ell k} \in \mathbb{R}^{F_{\ell-1} \times F_\ell}$ is the matrix collecting the kth coefficient of all filters in the bank, $[\mathbf{H}_{\ell k}]_{gf} = h_{\ell k}^{fg}$. In what follows, however, we choose the description in (25)–(26) which emphasizes the role of each graph filter, allowing us to better focus on the interaction between filters and nonlinearities. For ease of exposition, we assume that at each layer each feature is processed by F filters. This means that the first and last layer contain F filters whereas the remaining intermediate layers contain F^2 filters. In any case, the results derived here extend to intermediate layers with varying number of features in a straightforward manner (see proofs in the Appendix).

We emphasize that the nonlinear operation in (26) is applied to each entry of $\mathbf{z}_\ell^f$ individually. We further assume that the nonlinearity is normalized Lipschitz so that for all $a, b \in \mathbb{R}$,

$$|\sigma(b) - \sigma(a)| \leq |b - a|. \tag{28}$$

Asides from the input $\mathbf{x}$, the GNN's output depends on the filters $\mathbf{h}_\ell^{fg}$ and the graph $\mathbf{S}$. We interpret a GNN as a transform defined by the filter coefficients that we can apply on *any graph* to any

signal defined on the graph. Define then the map

$$\Phi(\mathbf{S}, \mathbf{x}) \;=\; \mathbf{x}_L^1 \tag{29}$$

to represent the outcome of applying (25)–(26) on graph $\mathbf{S}$ to input signal $\mathbf{x} = \mathbf{x}_0^1$. Our goal is to study the stability of the operator $\Phi(\mathbf{S}, \cdot)$ with respect to perturbations of the graph $\mathbf{S}$.

Remark 3 (Graph neural networks): We consider GNNs defined by equations (25)–(26). The literature includes several different proposed architectures such as ChebNets [11], GCNs [12] or Selection GNNs [13]. The model in equations (25)–(26) is the one that appears in [13] and it could be therefore interpreted as a particular choice. However, it is known that all of the architectures in [11]–[13] can be equivalently described by the GNN model in equations (25)–(26) [34], [35]. Therefore, the results that we derive can be applied to all of these architectures, which are formally known as graph *convolutional* neural networks. For a comprehensive framework of convolutional as well as non-convolutional graph neural networks, as well as a proof of equivalence between architectures, see [36].

A. Permutation Equivariance of GNNs

The superior performance of graph neural networks (GNNs) can be explained by the use of filter banks and nonlinearities to successfully process high-eigenvalue frequencies in a stable manner. An arbitrary GNN $\Phi(\mathbf{S}, \cdot)$ with L layers, over a graph representation $\mathbf{S}$, is defined by (25)–(26) for $\ell = 1, \ldots, L$. GNNs retain the two fundamental properties of linear filters. Namely, permutation equivariance and stability.

Proposition 2 (GNN permutation equivariance): Consider graph shifts $\mathbf{S}$ and $\hat{\mathbf{S}} = \mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P}$ for some permutation matrix $\mathbf{P} \in \mathcal{P}$ [cf. (4)]. Given a bank of filters $\{\mathbf{h}_\ell^{fg}\}$ for each layer $\ell = 1, \ldots, L$ and a pointwise nonlinearity σ, define a GNN Φ [cf. (25)–(26)]. Then, for any pair of corresponding graph signals $\mathbf{x}$ and $\hat{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{x}$ used as input to the GNN it holds that

$$\Phi(\hat{\mathbf{S}}, \hat{\mathbf{x}}) = \mathbf{P}^{\mathsf{T}}\Phi(\mathbf{S}, \mathbf{x}). \tag{30}$$

Proof: See Appendix D. ∎

Proposition 2 states that GNNs retain the permutation equivariance inherited from graph filters [cf. Prop 1], so that graph signal processing with GNNs is independent of node relabelings. To study stability of the GNN operator in (29), we therefore want to consider measures of proximity that are impervious to permutations. To that end we define an operator distance modulo permutation as follows.

Definition 5 (Operator Distance Modulo Permutation): Given operators $\Psi : \mathbb{R}^N \to \mathbb{R}^N$ and $\hat{\Psi} : \mathbb{R}^N \to \mathbb{R}^N$ we define their operator distance modulo permutation as

$$\left\|\Psi - \hat{\Psi}\right\|_{\mathcal{P}} \;=\; \min_{\mathbf{P}\in\mathcal{P}} \max_{\mathbf{x}:\|\mathbf{x}\|=1} \left\|\mathbf{P}^{\mathsf{T}}\Psi(\mathbf{x}) - \hat{\Psi}(\mathbf{P}^{\mathsf{T}}\mathbf{x})\right\| \tag{31}$$

where $\mathcal{P}$ is the set of $N \times N$ permutation matrices (cf. (4)) and where $\|\cdot\|$ stands for the ℓ_2-norm.

The operator distance in (31) compares operators Ψ and $\hat{\Psi}$ when the same permutations are applied at their respective inputs and output, and it is a generalization of Def. 1 to (nonlinear) operators. GNNs are insensitive to permutations, as shown by Prop. 2, and thus we have $\left\|\Phi - \hat{\Phi}\right\|_{\mathcal{P}} = 0$. The distance in (31) is thus a measure of how far from a permutation the operators are.

B. Stability of GNNs to Perturbations of the Graph

The stability of GNNs is inherited from that of the graph filters that conform the filter bank used in (25). The hyperparameters of the GNN further impact the stability.

Theorem 4 (GNN Stability): Let $\mathbf{S}$ and $\hat{\mathbf{S}}$ be GSOs related by perturbation matrix $\mathbf{E}$ [cf. (7) or (18)] such that $\|\mathbf{E}\| \leq \varepsilon$. Given a bank of filters $\{\mathbf{h}_\ell^{fg}\}$ such that $|h_\ell^{fg}(\lambda)| \leq 1$ [cf. (11)] and a pointwise nonlinearity σ that is Lipschitz continuous (28), define GNNs $\Phi(\mathbf{S}, \cdot)$ and $\Phi(\hat{\mathbf{S}}, \cdot)$ [cf. (25)–(26)]. If the corresponding filter banks satisfy $\|\mathbf{H}_\ell^{fg}(\mathbf{S}) - \mathbf{H}_\ell^{fg}(\hat{\mathbf{S}})\|_{\mathcal{P}} \leq \Delta\varepsilon$, then it holds that

$$\|\Phi(\mathbf{S}, \cdot) - \Phi(\hat{\mathbf{S}}, \cdot)\|_{\mathcal{P}} \leq \Delta L F^{L-1}\varepsilon + \mathcal{O}(\varepsilon^2) \tag{32}$$

for a GNN with a single input feature, a single output feature and F features in each hidden layer.

Proof: See Appendix E. ∎

Theorem 4 establishes how the stability of the filters Δ is affected by the hyperparameters of the GNN architecture. More specifically, we see that the stability gets degraded linearly with the number of layers L, and exponentially with the number of features F (with an exponent controlled by L). In essence, the deeper a GNN is, the less stable it is. We note that this causes the bound to be quite loose, as is also evidenced in Section V. However, we see that the result is still linear in the size of the perturbation ε and in the stability constant Δ of the filters. This stability constant depends on the perturbation model under consideration (either absolute –Section II-B– or relative –Section II-C–) and on the Lipschitz condition on the graph filters (either Lipschitz –Def. 2– or integral Lipschitz –Def. 4–), as determined next.

Proposition 3: Under the conditions of Theorem 4, with $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathsf{H}}$, consider the following models.

1) If matrix $\mathbf{E} = \mathbf{U}\mathbf{M}\mathbf{U}^{\mathsf{H}}$ models absolute perturbations [cf. (7)], and the filters are Lipschitz (Def. 2) we have
$$\mathbf{\Delta} = C(1 + \delta\sqrt{N}) \tag{33}$$
with $\delta = (\|\mathbf{U} - \mathbf{V}\| + 1)^2 - 1$.
2) If matrix $\mathbf{E} = \mathbf{U}\mathbf{M}\mathbf{U}^{\mathsf{H}}$ models relative perturbations (Def. 3), and the filters are integral Lipschitz (Def. 4),
$$\mathbf{\Delta} = 2\,C(1 + \delta\sqrt{N}) \tag{34}$$
holds with $\delta = (\|\mathbf{U} - \mathbf{V}\| + 1)^2 - 1$.
3) If matrix $\mathbf{E}$ models relative perturbations (Def. 3) and satisfies (23), and the filters are integral Lipschitz (Def. 4),
$$\mathbf{\Delta} = 2\,C. \tag{35}$$

Proof: Follows directly from Theorem 4 in combination with Thms. 1, 2 and 3. These theorems establish the conditions and the corresponding values of Δ. ∎

The results of Theorem 4 in combination with Prop. 3 show that: (i) the use of Lipschitz filters lead to stable GNNs under absolute perturbations, and (ii) the use of integral Lipschitz filters lead to stable GNNs under relative perturbations. In all

cases, Theorem 4 establishes that GNNs $\mathbf{\Phi}(\mathbf{S}, \mathbf{x})$ are Lipschitz stable with respect to the changes in the underlying graph support $\mathbf{S}$, and not with respect to changes in the input $\mathbf{x}$.

The stability constant in (32) in combination with the value of Δ in (33) or (34) consists of the product of three terms. One given by the filter's (integral) Lipschitz constant, C, one given by the number of filters and layers in the GNN, LF^{L-1}, and one containing the eigenvector misalignment constant $(1 + \delta\sqrt{N})$. The one related to the GNN architecture is just a consequence of perturbations propagating across different filters. The other two terms represent different fundamental facets of GNNs. The (integral) Lipschitz constant C is a property of the filters which is up for choice during filter design, or, perhaps more likely, expected as an outcome of the training process. The term $(1 + \delta\sqrt{N})$ is a property of the family of perturbations $\mathbf{E}$ that are admissible which is an inherent property of the type of perturbations we expect to see in a specific problem. It is important to remark that we can affect C by designing or learning proper filters but we cannot affect δ. The latter is not a property of the filter, but a property of the perturbation $\mathbf{E}$. The important point is that, regardless of δ, judicious choice of filter coefficients $\mathbf{h}$, affects the (integral) Lipschitz constant C and allows control of the stability of the graph filters that define a GNN.

The value of ε in model (i) of Prop. 3 represents the absolute perturbation distance [cf. (7)] between shifts $\mathbf{S}$ and $\hat{\mathbf{S}}$ and as such, is independent of the actual particularities of the graph under study (a fixed value of ε would mean a different perturbation level for graphs that have very different edge weights). Likewise, the value of C is given by the Lipschitz constant of the filters. The higher the value of C, the more selective the filters can be (the more narrow they can be), but the more unstable the GNNs become. Finally, the value of δ accounts for the eigenvector misalignment between the absolute error matrix $\mathbf{E}$ and the shift $\mathbf{S}$, which indicates the impact on the spectrum basis by the perturbation, and affects the stability bound by a value dependent on the number of nodes (the larger the graph, the more a change in the spectrum basis affects stability).

With respect to model (ii) of Prop. 3 we observe that now ε represents the relative distance between $\mathbf{S}$ and its perturbation $\hat{\mathbf{S}}$ [cf. (18)], meaning that a fixed ε represents the same level of perturbation for any possible reweighing of the difference $\alpha(\mathbf{S} - \hat{\mathbf{S}})$, $\alpha \in \mathbb{R}$. The value of C, in this case, represents the integral Lipschitz constant of the filters (cf. Def. 4). Integral Lipschitz filters, however, can be made arbitrarily selective near $\lambda \approx 0$, irrespective of the value of C, allowing for perfect discrimination of features around it, without affecting the overall stability. In integral Lipschitz filters, the value of C determines the smallest eigenvalue for which the filter response becomes (approximately) flat, and hence loses discriminative power. A high value of C would allow for greater selectivity in higher-eigenvalue frequencies, but at the expense of stability. With respect to δ, the same analysis as for model (i) holds, except that in this case, the eigenvectors $\mathbf{U}$ correspond to the *relative* error matrix $\mathbf{E}$. We also note that the presence of δ causes the bound to be quite loose for large values of N.

To overcome the degradation of the stability with the size of the graph, we propose model (iii) of Prop. 3. In this model, where ε measures the relative perturbation distance and C the integral Lipschitz constant, the family of admissible perturbations has been restricted to those that satisfy the structural constraint (23). Admissible perturbations are now those that are either dilations or contractions of the edge weights of $\mathbf{S}$. Dilations and contractions can be different for different nodes but cannot be a mix of dilation and contraction in different parts of the graph. We remark that if the structural constraint is satisfied, then the stability can be controlled by determining the integral Lipschitz constant of the filters, for any graph. However, for some specific families of graphs, where we have information on how the eigenvectors change with a given perturbation size, we can improve on the result by relaxing the structural constraint. This is the case of [3], where extraneous geometric information (Euclidean space) is leveraged to quantify the impact of the perturbation (diffeomorphism) on the spectrum basis.

IV. Discussions

From the analysis of model (i) in Prop. 3 we concluded that Lipschitz filters are stable under absolute perturbations, but the stability presents a trade-off with the selectivity of the filters (the more stable the GNN is, the less selective are the filters that compose it). Moreover, we commented that the absolute perturbation model presents certain limitations by not taking into account the underlying graph support.

Under a relative perturbation model, integral Lipschitz filters can be made arbitrarily selective near $\lambda \approx 0$ without sacrificing stability. Therefore, in order to discriminate among signals with frequency content in high values of λ we need to spill the information into lower-eigenvalue frequencies, which is easily achieved by the mixing effect of the nonlinearities employed. The following discussion illustrates the intricacies of the stability results put forward in Theorem 4 and Prop. 3.

Suppose that we have shift operators $\mathbf{S}$ and $\hat{\mathbf{S}}$ where the latter is a simple scaling of the former by a factor $(1 + \varepsilon)$

$$\hat{\mathbf{S}} = (1 + \varepsilon)\mathbf{S}. \tag{36}$$

The graph dilation in (36) produces a graph in which all edges are scaled by a $(1 + \varepsilon)$ factor. This is a perturbation model of the form in (17) with $\mathbf{E} = (\varepsilon/2)\mathbf{I}$. We consider that $\varepsilon \approx 0$ in which case the graph dilation produces a minimal modification of the graph. Note that, for such a perturbation, we have $\delta = 0$ in model (ii) and it also satisfies the structural constraint (23) of model (iii), so that both models are applicable here.

Suppose now that we are given a set of filter coefficients $\mathbf{h}$ and that we consider the filter $\mathbf{H}(\mathbf{S})$ implemented on GSO $\mathbf{S}$ vis-à-vis the filter $\mathbf{H}(\hat{\mathbf{S}})$ implemented on another GSO $\hat{\mathbf{S}}$ [cf. (2)–(3)]. Given that the graph perturbation is inconsequential we would expect the filter differences to be inconsequential as well. Theorem 2 states that if the filters are integral Lipschitz this is true but if they are simply Lipschitz this need not be true. To understand this we look at the differences between the spectra of $\mathbf{S}$ and $\hat{\mathbf{S}}$.

Given that $\mathbf{S}$ and $\hat{\mathbf{S}}$ are related by a scaling, they share the same eigenvectors and the scaling is translated to the eigenvalues. Thus, if $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathsf{H}}$ is the eigenvector decomposition of $\mathbf{S}$ [cf. (1)], the eigenvector decomposition of $\hat{\mathbf{S}}$ is

$$\hat{\mathbf{S}} = \mathbf{V}\left[(1 + \varepsilon)\mathbf{\Lambda}\right]\mathbf{V}^{\mathsf{H}}. \tag{37}$$

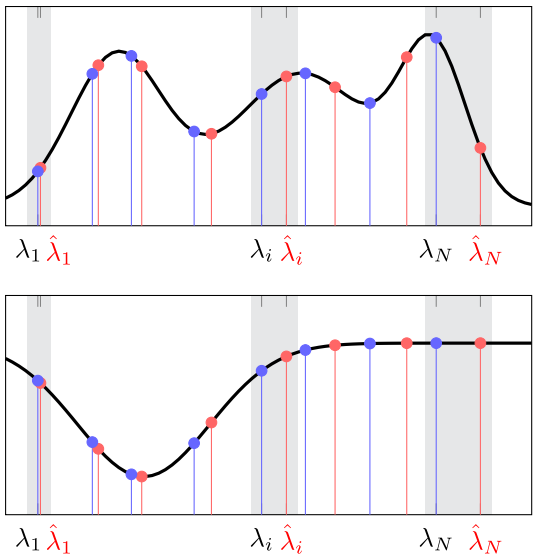


Fig. 2. Stability of graph filters. We observe that, for small values of λ, the difference between λ_i (in blue) and $\hat{\lambda}_i$ (in red) is small, whereas for large λ this becomes much larger. (top) When using a Lipschitz filter [cf. (13)], we observe that for low frequencies, the response of the filter is very similar when instantiated on either λ_i or $\hat{\lambda}_i$; however, for large frequencies, the difference becomes much larger, and thus a small change in the eigenvalues, leads to a big change of the filter response. (bottom) In the case of integral Lipschitz filters [cf. (19)], the effect on high frequencies is mitigated, by forcing the filter to be nearly constant at these frequencies, so that, when evaluated at eigenvalues that are far away, the filter response is still almost the same, guaranteeing stability.

As per (37), the eigenvalues of $\hat{\mathbf{S}}$ are the eigenvalues of $\mathbf{S}$ scaled by a factor $(1+\varepsilon)$. Thus, the effect of the dilation in (36) on a filter with frequency response $h(\lambda)$ is that instead of instantiating the response at eigenvalues λ_i we instantiate it at eigenvalues $(1+\varepsilon)\lambda_i$. Consequently the response values that we expect to be $h(\lambda_i)$ if the filter is run on $\mathbf{S}$ actually turn out to be $h((1+\varepsilon)\lambda_i)$ if the filter is run on $\hat{\mathbf{S}}$. This observation is the core argument in the proof of Theorem 2 and motivates the important observations that we discuss next.

Graph perturbations and filter perturbations Fig. 2 illustrates the effect of the dilation in (36) on a Lipschitz (top) and integral Lipschitz filter (bottom). The difference in the positions between eigenvalues is given by $\hat{\lambda}_i - \lambda_i = \varepsilon\lambda_i$, and as such, depends on the value of the specific eigenvalue λ_i. For low-eigenvalue frequencies λ_i the dilation results in a small perturbation of the eigenvalues. If the change in eigenvalues is small the change in the filter's response from $h(\lambda_i)$ to $h(\hat{\lambda}_i)$ is small for both filters. For large eigenvalues the difference $\hat{\lambda}_i - \lambda_i = \varepsilon\lambda_i$ grows large. For Lipschitz filters a large difference in the arguments may translate into a large difference in the instantiated values of frequency responses $h(\hat{\lambda}_i)$ and $h(\lambda_i)$,

$$|h(\hat{\lambda}_i) - h(\lambda_i)| \approx |\hat{\lambda}_i - \lambda_i| = \varepsilon\lambda_i. \tag{38}$$

This explains the filter's instability. A small graph perturbation may result in a large filter perturbation at high-eigenvalue frequencies. For integral Lipschitz filters, on the other hand, changes in the frequency response must taper off as λ grows.

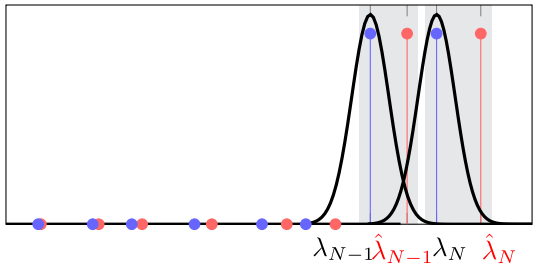


Fig. 3. High frequency feature extraction. We illustrate two sharp filters designed to successfully extract high frequency features located at λ_{N-1} and λ_N. However, when the graph is slightly perturbed, which results in large changes in high frequency eigenvalues, the designed filters are no longer able to extract these features, now located at $\hat{\lambda}_{N-1}$ and $\hat{\lambda}_N$, since they have moved out of the narrow pass band of the filter.

Thus, even though there may be a large variation in the eigenvalues the instances of the frequency responses are close

$$|h(\hat{\lambda}_i) - h(\lambda_i)| \approx \frac{|\hat{\lambda}_i - \lambda_i|}{|\hat{\lambda}_i + \lambda_i|/2} = \frac{2\varepsilon}{2-\varepsilon} \approx \varepsilon. \tag{39}$$

This explains the filter's stability. No matter how large the eigenvalues are, a small perturbation of the graph results in a small perturbation of the graph filter. Theorem 2 shows that this is true for arbitrary relative perturbations.

Graph perturbations and feature identification: There is an obvious cost we pay for the stability of integral Lipschitz filters: they are unable to discriminate high-eigenvalue frequencies. The graph dilation example shows that this is not a limitation of the analysis. It is impossible to have a filter that is both stable and able to isolate high-eigenvalue features because small graph perturbations can result in large eigenvalue perturbations. This is a major drawback of linear graph filters in the extraction of features from graph signals. To illustrate this drawback suppose we have graph signals $\mathbf{x}_1 = \mathbf{v}_N$ and $\mathbf{x}_2 = \mathbf{v}_{N-1}$ and we want to design graph filters to discriminate between the two. The graph frequency domain representation of these two signals on the graph $\mathbf{S}$ are shown in Fig. 3. For us to discriminate between $\mathbf{x}_1 = \mathbf{v}_N$ and $\mathbf{x}_2 = \mathbf{v}_{N-1}$ we need filters centered at frequencies λ_N and λ_{N-1}. These filters must have sharp transitions so that the filter isolating $\mathbf{x}_1 = \mathbf{v}_N$ does not let the signal $\mathbf{x}_2 = \mathbf{v}_{N-1}$ pass and, conversely, the filter isolating $\mathbf{x}_2 = \mathbf{v}_{N-1}$ does not let the signal $\mathbf{x}_1 = \mathbf{v}_N$. Yet, if these filters are sharp on large eigenvalues, they will be unstable. More specifically, let $\hat{\lambda}_N = (1+\varepsilon)\lambda_N$ be the eigenvalue associated to $\mathbf{x}_1 = \mathbf{v}_N$ in the perturbed graph, and $\hat{\lambda}_{N-1} = (1+\varepsilon)\lambda_{N-1}$ be the one associated to $\mathbf{x}_2 = \mathbf{v}_{N-1}$. Now, since the filters were designed to be sharp around λ_N and λ_{N-1}, but the perturbed eigenvalues $\hat{\lambda}_N$ and $\hat{\lambda}_{N-1}$ are far from these (at points where the filter response is virtually zero) the filter fails to adequately recover $\mathbf{x}_1$ and $\mathbf{x}_2$ in the perturbed graph. See Fig. 3 for an illustration of the instability effect at large eigenvalues.

Pointwise nonlinearities: So far, we have observed that stable filters require a flat response on high-eigenvalue frequencies, but that this inevitably prevents them from discriminating between features located at these frequencies. This illustrates an inherent, insurmountable limitation of linear information processing

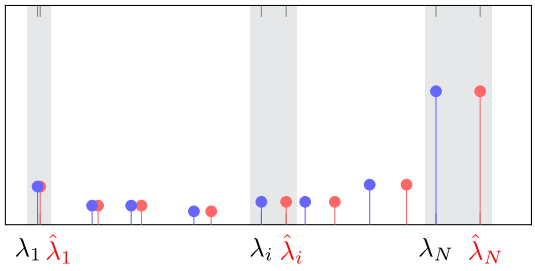


Fig. 4. Effect of pointwise nonlinearity. Let $\mathbf{x} = \mathbf{v}_N$ be the graph signal with a frequency response $\tilde{\mathbf{x}}$ given by $\tilde{x}_N = 1$ and $\tilde{x}_i = 0$ for all $i = 1, \ldots, N-1$. Signal $\mathbf{x}$ has a single nonzero value located at the highest frequency, making it impossible to be extracted with a stable linear filter. When applying a nonlinearity to this signal, we observe that nonzero frequency components arise throughout the spectrum, spilling the information contained in the highest frequency into lower frequencies. This facilitates the use of a bank of stable linear filters to successfully collect this information at lower frequencies.

schemes. Neural networks introduce pointwise nonlinearities to the processing pipeline, as a computationally straightforward means of discriminating information located at large eigenvalues. The basic effect of these nonlinearities is to cause a spillage of information throughout the frequency band, see Fig. 4. This spillage of information into smaller eigenvalues allows for a stable filter to accurately discriminate between them, since information at these frequencies does not get severely affected by perturbations. However, since the energy in smaller eigenvalues is usually less than the energy still found at larger ones, and since it is also spread through a wide band of frequencies, the use of a bank of linear filters becomes a sensitive idea to better capture this spillage. Therefore, the use of banks of linear filters in combination with pointwise nonlinearities allows for information processing architectures that are able to capture high-eigenvalue frequency content in a stable fashion.

Permutation Equivariance: The permutation equivariance stated in Prop. 2 shows that the features that are learned by a GNN are independent of the labeling of the graph. But permutation equivariance is also important because it means that GNNs exploit internal signal symmetries as we illustrate in Fig. 5. The graphs in Figs. 5(a) and 5(b) are the same, as indicated by the integer labels. The signals in Figs. 5(a) and 5(b) are different, as indicated by different colors. However, it is possible to permute the graph onto itself to make the signals match – rotate 180° degrees and pull it inside out (Fig. 5 c). It then follows from Prop. 2 that the output of a GNN applied to the signal on the left (5a) is a corresponding permutation of the output of the same GNN applied to the signal on the right (5b). This is beneficial because we can learn to process the signal on (5a) from seeing examples of the signal on (5b). We note that, while most graphs do not exhibit perfect symmetries, they might have (sub)structures that are close to permutations. Therefore, permutation equivariance shows the ability of GNNs to exploit these similarities.

V. Numerical Experiments

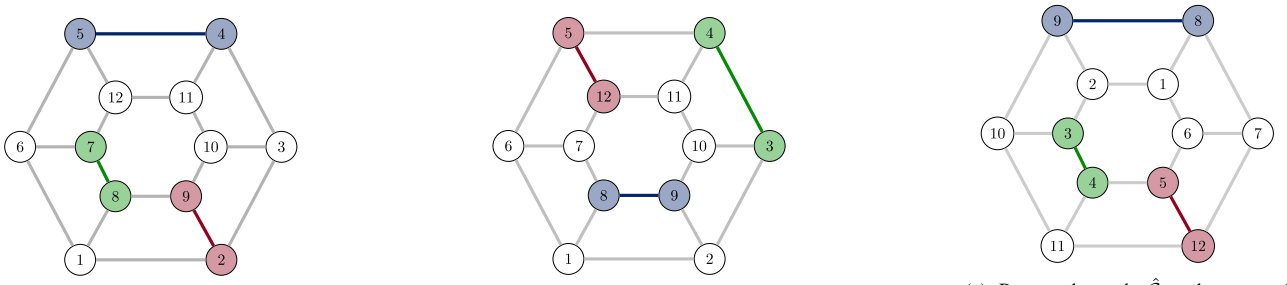
To illustrate the GNN stability results in a practical setting, we consider the problem of movie recommendation systems [37]. We describe the problem with a graph where each node is a movie, and each edge weight represents the rating similarity for each pair of movies. The information from each user is modeled as a graph signal, whereby the value assigned to each node represents the rating the user has given to each movie watched. The objective is to infer the rating a user would give to a specific, unseen movie, based on the ratings given to the other movies, and the rating similarities present in the graph structure. We carry out two experiments, the first one showing the stability under a synthetic, controlled relative perturbation; and the second one considering a more realistic perturbation arising from the error in estimating the underlying graph structure. The main objectives of this section are to illustrate how *loose* the bound actually is and also that GNNs using integral Lipschitz filters are more stable (which amounts to showing the effect of C).

Dataset: We use the MovieLens-100 k dataset [38]. This dataset contains $100,000$ ratings given by 943 users to some of the 1582 movies available. Ratings go from 1 indicating a disliked movie, to 5 indicating a liked movie. The movie *Star Wars* is selected as the target movie to estimate the rating, since it is the movie with the largest number of available ratings.

Graph signal processing formulation: We use 90% of the ratings as part of the training set in order to build the graph support. Each node in the graph is a movie, amounting to 1582 nodes. The edge weights are obtained by estimating the Pearson correlation coefficient between each pair of movies as in [37, eq. (6)], based on the ratings contained in the training set only. A 10 nearest-neighbor graph is built from these edge weights. Once the graph is built, we consider the users that have rated the target movie, which amounts to 583 users. Each of these users is considered as a graph signal, where each node value is the rating given to that movie. Movies not rated by each user are assigned a 0. The rating given to the target movie is extracted as a label, and zeroed out in the graph signal. This dataset of 583 graph signals and the corresponding labels is split into 90% for the training set and 10% for the testing set, with the training set further split into 10% for validation, and the rest for training.

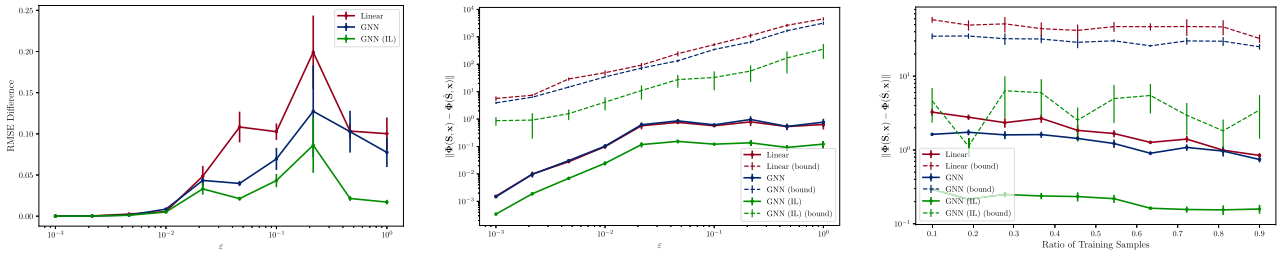
Architectures: We consider two GNN architectures as mapping parametrizations between the input graph signal (the ratings given to some of the movies) and the label (the rating given to the target movie). The architectures have a single-layer GNN (25)–(26) with $F_0 = 1$ input feature (the rating value) and $F_1 = 64$ output features, and 5 filter taps. The nonlinearity used is a ReLU. One of the architectures, labeled as 'GNN,' learns from the space of all graph filters, while another one, labeled as 'GNN (IL),' learns only integral Lipschitz filters. We compare these two architectures with a learned linear graph filter with 64 output features and 5 filter taps. All architectures have a local, linear readout layer mapping the 64 features at the target node to a single scalar that estimates the rating (i.e. a learnable 64×1 matrix which is equivalent to a second layer with $F_2 = 1$ and 1 filter tap).

Training and evaluation: We train all the architectures by minimizing a smooth $L1$ loss between the estimated rating at the output of the readout layer and the extracted label. We use an ADAM optimizer with learning rate 0.005 and forgetting factors 0.9 and 0.999. We train for 40 epochs with batches of size 5. The



(a) Graph $\mathcal{G}$ and signal $\mathbf{x}$ (b) Graph $\mathcal{G}$ and permuted signal $\mathbf{P}^{\mathsf{T}}\mathbf{x}$ (c) Permuted graph $\hat{\mathcal{G}}$ and permuted signal $\mathbf{P}^{\mathsf{T}}\mathbf{x}$

Fig. 5. Permutation equivariance of graph neural networks (GNNs). The output of a GNN is equivariant to graph permutations (Proposition 1). This not only means independence from labeling but it also shows that GNNs exploit internal signal symmetries. The signals on (a) and (b) are different signals on the same graph but they are permutations of each other – interchange inner and outer hexagons and rotate 180° [c.f. (c)]. A GNN would learn how to classify the signal in (b) from seeing examples of the signal in (a). Integers represent the labeling, while colors represent graph signal values.



(a) (b) (c)

Fig. 6. Movie Recommendation problem. The baseline evaluation performance (RMSE) is $0.84(\pm 0.15)$ for the linear architecture (Linear), $0.84(\pm 0.16)$ for the GNN that learned from the space of all graph filters (GNN), and $0.83(\pm 0.14)$ for the GNN that learned integral Lipschitz filters (GNN (IL)). Synthetic Experiment: (a) Change in evaluation measure (RMSE) due to synthetic changes in the underlying graph support, where we observe that the GNN (IL) is more stable than the GNN and the Linear architectures; (b) Change in the output of the GNN due to synthetic changes in the underlying graph support, we observe that the GNN (IL) is consistently more stable, and that the bounds are not tight. Estimation error experiment. (c) Changes in the output of the GNN due to changes in the estimation of the graph support, stemming from using different sizes of training set; again, we observe that GNN (IL) is consistently more stable than the other two architectures and that the bounds are not tight.

evaluation performance is the root mean squared error (RMSE) as is standard in the movie recommendation problem [37]. In all cases, we run 5 random dataset partitions, and report the average performance across these realizations, as well as the standard deviation.

Experiments: We run two experiments. For the first experiment, we consider synthetic relative perturbations, and analyze how the output of the GNN and the evaluation performance change under controlled perturbations of the graph support at test time. For the second experiment, we consider a real world perturbation stemming from different construction of the graph support. That is, note that the graph support is build out of the training set, so changing the training set would lead to different graph support, each reflecting a different estimation of the Pearson coefficient. In particular, smaller training sets would lead to larger estimation error, and thus, by changing the ratio of the training/testing set split, we can adjust the estimation error. As a matter of fact, we note in practice that using smaller sets to build the graph leads to a larger relative perturbation. In the second experiment we analyze the stability of the GNNs for different values of the training set ratio, analyzing the usefulness of stable architectures in a real world setting with inference estimation errors [39].

Synthetic experiment: We generate a random perturbation matrix $\mathbf{E}$ such that $\|\mathbf{E}\| \leq \varepsilon$ and (23) are satisfied. We do so by generating a diagonal matrix $\mathbf{E}$, with diagonal elements drawn uniformly at random from the interval $[(1-\varepsilon)\varepsilon, \varepsilon])$. Note that such a perturbation is not a simple edge dilation like the one discussed in Section IV. We then build the perturbed matrix $\hat{\mathbf{S}} = \mathbf{S} + \mathbf{ES} + \mathbf{SE}$. Note that we do not recompute the 10 nearest neighbors. We control the perturbation size ε and simulate it from 10^{-3} to 1. The change in the evaluation measure (the change in the RMSE) can be found in Fig. 6(a). We observe that, for small ε, there is virtually no change in the output between all three architectures, but as ε grows, the change in the GNN with integral Lipschitz filters is smaller than the change in both the linear and the GNN architectures. This evidences that the GNN with integral Lipschitz filters is indeed more stable. In Fig. 6(b) we show specifically the change in the output of the GNN layer caused by the perturbation. We also show the the bounds. First, we note that the GNN that learned integral Lipschitz filters is consistently more stable. We also note that the bounds are not tight bounds, essentially because the bound on the eigenvectors is valid for all graphs and thus is not tight.

Estimation error experiment: In this last experiment, we consider a more realistic perturbation. We consider architectures

trained on a graph based on Pearson correlations estimated from a 90% split of the training set. Then, at test time, we consider architectures running on graphs with Pearson correlations estimated from smaller training sets, ranging from 10% to 90%. Since the number of training sets are smaller, then the estimation error of the graph is larger. This is a scenario that arises when the underlying graph support is not known and needs to be estimated (so we can consider $\mathbf{S}$ to be the true support, and $\hat{\mathbf{S}}$ to be the estimation) [39]. The change at the output of the GNN layer is shown in Fig. 6(c). We see that the GNN with integral Lipschitz filters is approximately one order of magnitude more stable than the GNN trained with arbitrary graph filters. This one, in turn, is slightly more stable than the linear architecture. Likewise, we show the bounds and see that they are not tight.

VI. Conclusion

We focused on the impact that changes in the underlying topology have on the output of a GNN. First, we studied changes brought by permutations. We proved that GNNs are permutation equivariant, and that this implies that they effectively exploit the topological symmetries present in the underlying graph. Then, we discussed the absolute perturbation model existing in the literature, and proved that GNNs composed of Lipschitz filters are stable. However, not only the absolute perturbation model ignores the particularities of the underlying graph, but also the stability comes at the expense of the discriminative power of the filters (i.e. the more stable, the less discriminative). We thus proposed a relative perturbation model and proved that filters used in GNNs need be *integral* Lipschitz for the resulting architecture to be stable. Integral Lipschitz filters can be made arbitrarily selective around low-eigenvalue frequencies, but need to have a flat response in high-eigenvalue frequencies, precluding accurate discrimination of information located in this band. We show that the frequency mixing effect of nonlinearities succeeds in spreading the information throughout the frequency spectrum, and thus allowing for accurate discrimination of information located at all frequencies. In essence, superior performance of GNNs can be explained by the fact that they are both stable and discriminative architectures, whereas linear graph filters can only satisfy one of these properties. We illustrated the discriminability and stability properties of both GNNs and graph filters in a movie recommendation problem. It was observed in the experiments that the bounds are not tight, and thus they can be improved. One of the reasons for this lack of tightness is that the bound in Theorem 4 holds for all possible perturbations, resulting in a rather large value of the vector misalignment constant. This bound can certainly be improved if more specific perturbation models are studied, giving raise to particular bounds for the constant. This is envisioned as a future area of research, where the bounds provided herein are improved for specific applications on specific graphs. Likewise, Theorem 4 holds for perturbations that have the same number of nodes as the original graph. Extending this result to graphs of different size is an active area of research.

Appendix A
Permutation Equivariance of Graph Filters

Proof of Prop. 1: A permutation matrix $\mathbf{P} \in \mathcal{P}$ is an orthogonal matrix, $\mathbf{P}^{\mathsf{T}}\mathbf{P} = \mathbf{P}\mathbf{P}^{\mathsf{T}} = \mathbf{I}$, from where it follows that powers $\hat{\mathbf{S}}^k$ of a permuted shift operator are permutations of the respective shift operator powers $\mathbf{S}^k$

$$\hat{\mathbf{S}}^k = (\mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P})^k = \mathbf{P}^{\mathsf{T}}\mathbf{S}^k\mathbf{P}. \tag{40}$$

Substituting this fact in the definition of the permuted graph filter $\mathbf{H}(\hat{\mathbf{S}})$ in (3) yields

$$\mathbf{H}(\hat{\mathbf{S}}) = \sum_k h_k \left(\mathbf{P}^{\mathsf{T}}\mathbf{S}^k\mathbf{P}\right) = \mathbf{P}^{\mathsf{T}}\left(\sum_k h_k\mathbf{S}^k\right)\mathbf{P}. \tag{41}$$

In the last equality the sum is the filter $\mathbf{H}(\mathbf{S}) = \sum_k h_k\mathbf{S}^k$ as defined in (2). We can then write $\mathbf{H}(\hat{\mathbf{S}}) = \mathbf{P}^{\mathsf{T}}\mathbf{H}(\mathbf{S})\mathbf{P}$ and use this fact to express application of the permuted filter $\mathbf{H}(\hat{\mathbf{S}})$ to the permuted signal $\hat{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{x}$ as

$$\hat{\mathbf{z}} = \mathbf{H}(\hat{\mathbf{S}})\hat{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{H}(\mathbf{S})\mathbf{P}\mathbf{P}^{\mathsf{T}}\mathbf{x} \tag{42}$$

Since $\mathbf{P}$ is orthogonal we have that $\mathbf{P}\mathbf{P}^{\mathsf{T}} = \mathbf{I}$. Substituting this into the right hand side of (42), the result in (5) follows. ■

Appendix B
Stability Under Absolute Perturbations

Lemma 1: Let $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathsf{H}}$ and $\mathbf{E} = \mathbf{U}\mathbf{M}\mathbf{U}^{\mathsf{H}}$ such that $\|\mathbf{E}\| \leq \varepsilon$. For any eigenvector $\mathbf{v}_i$ of $\mathbf{S}$ it holds that

$$\mathbf{E}\mathbf{v}_i = m_i\mathbf{v}_i + \mathbf{E}_U\mathbf{v}_i \tag{43}$$

with $\|\mathbf{E}_U\| \leq \varepsilon\delta$, where $\delta = (\|\mathbf{U} - \mathbf{V}\|^2 + 1)^2 - 1$.

Proof: Start by writing the error matrix $\mathbf{E}$ as

$$\mathbf{E} = \mathbf{E}_V + \mathbf{E}_U \tag{44}$$

$$\mathbf{E}_V = \mathbf{V}\mathbf{M}\mathbf{V}^{\mathsf{H}} \tag{45}$$

$$\mathbf{E}_U = (\mathbf{U} - \mathbf{V})\,\mathbf{M}\,(\mathbf{U} - \mathbf{V})^{\mathsf{H}} + \mathbf{V}\mathbf{M}\,(\mathbf{U} - \mathbf{V})^{\mathsf{H}} + (\mathbf{U} - \mathbf{V})\,\mathbf{M}\mathbf{V}^{\mathsf{H}}. \tag{46}$$

We see that $\mathbf{E}_V\mathbf{v}_i = m_i\mathbf{v}_i$ since $\mathbf{v}_i$ is an eigenvector of $\mathbf{E}_V$. Next, note that, since $\|\mathbf{E}\| \leq \varepsilon$, then $\|\mathbf{M}\| \leq \varepsilon$, so that

$$\begin{aligned}\|\mathbf{E}_U\| &\leq \left\|(\mathbf{U} - \mathbf{V})\mathbf{M}(\mathbf{U} - \mathbf{V})^{\mathsf{H}}\right\| + \left\|\mathbf{V}\mathbf{M}(\mathbf{U} - \mathbf{V})^{\mathsf{H}}\right\| + \left\|(\mathbf{U} - \mathbf{V})\mathbf{M}\mathbf{V}^{\mathsf{H}}\right\| \\ &\leq \|\mathbf{U} - \mathbf{V}\|^2\|\mathbf{M}\| + 2\|\mathbf{U} - \mathbf{V}\|\|\mathbf{V}\|\|\mathbf{M}\| \\ &\leq \varepsilon\|\mathbf{U} - \mathbf{V}\|^2 + 2\varepsilon\|\mathbf{U} - \mathbf{V}\| \\ &= \varepsilon\left((\|\mathbf{U} - \mathbf{V}\|^2 + 1)^2 - 1\right) = \varepsilon\delta\end{aligned} \tag{47}$$

which completes the proof. ■

Proof of Theorem 1: Since graph filters are permutation equivariant (Prop. 1), we can assume, without loss of generality, that $\mathbf{P}_0 = \mathbf{I}$ in (7), writing $\hat{\mathbf{S}} = \mathbf{S} + \mathbf{E}$. Let us start by computing the first order expansion of $(\mathbf{S} + \mathbf{E})^k$

$$(\mathbf{S} + \mathbf{E})^k = \mathbf{S}^k + \sum_{r=0}^{k-1}\mathbf{S}^r\mathbf{E}\mathbf{S}^{k-r-1} + \mathbf{C} \tag{48}$$

with $\mathbf{C}$ such that $\|\mathbf{C}\| \leq \sum_{r=2}^{k} \binom{k}{r} \|\mathbf{E}\|^r \|\mathbf{S}\|^{k-r}$. Using this first-order approximation back in (2), we get

$$\mathbf{H}(\hat{\mathbf{S}}) - \mathbf{H}(\mathbf{S}) = \sum_{k=0}^{\infty} h_k \sum_{r=0}^{k-1} \mathbf{S}^r \mathbf{E} \mathbf{S}^{k-r-1} + \mathbf{D} \tag{49}$$

with $\mathbf{D}$ such that $\|\mathbf{D}\| = \mathcal{O}(\|\mathbf{E}\|^2)$ since the coefficients $\{h_k\}_{k=0}^{\infty}$ of the filter stem from the power series expansion of the analytic function h which has bounded derivatives.

Next, consider an arbitrary graph signal $\mathbf{x}$ with finite energy $\|\mathbf{x}\| < \infty$ that has a GFT given by $\tilde{\mathbf{x}} = [\tilde{x}_1, \ldots, \tilde{x}_N]^{\mathsf{T}}$ so that $\mathbf{x} = \sum_{i=1}^{N} \tilde{x}_i \mathbf{v}_i$ for $\{\mathbf{v}_i\}_{i=1}^{N}$ the eigenvector basis of the GSO,

$$\left[\mathbf{H}(\hat{\mathbf{S}}) - \mathbf{H}(\mathbf{S})\right] \mathbf{x} = \sum_{i=1}^{N} \tilde{x}_i \mathbf{D} \mathbf{v}_i + \sum_{i=1}^{N} \tilde{x}_i \sum_{k=0}^{\infty} h_k \sum_{r=0}^{k-1} \mathbf{S}^r \mathbf{E} \mathbf{S}^{k-r-1} \mathbf{v}_i. \tag{50}$$

Let us focus on the second term of the sum in (50). It is immediate that $\mathbf{S}^{k-r-1}\mathbf{v}_i = \lambda_i^{k-r-1}\mathbf{v}_i$, so that

$$\sum_{i=1}^{N} \tilde{x}_i \sum_{k=0}^{\infty} h_k \sum_{r=0}^{k-1} \mathbf{S}^r \mathbf{E} \mathbf{S}^{k-r-1} \mathbf{v}_i = \sum_{i=1}^{N} \tilde{x}_i \sum_{k=0}^{\infty} h_k \sum_{r=0}^{k-1} \lambda_i^{k-r-1} \mathbf{S}^r \mathbf{E} \mathbf{v}_i \tag{51}$$

Now, using Lemma 1 in (51) yields two terms

$$\sum_{i=1}^{N} \tilde{x}_i \sum_{k=0}^{\infty} h_k \sum_{r=0}^{k-1} \lambda_i^{k-r-1} \mathbf{S}^r \mathbf{E} \mathbf{v}_i \tag{52}$$

$$= \sum_{i=1}^{N} \tilde{x}_i \sum_{k=0}^{\infty} h_k \sum_{r=0}^{k-1} \lambda_i^{k-r-1} \mathbf{S}^r m_i \mathbf{v}_i \tag{53}$$

$$+ \sum_{i=1}^{N} \tilde{x}_i \sum_{k=0}^{\infty} h_k \sum_{r=0}^{k-1} \lambda_i^{k-r-1} \mathbf{V} \mathbf{\Lambda}^r \mathbf{V}^{\mathsf{H}} \mathbf{E}_U \mathbf{v}_i. \tag{54}$$

For (53) we note that $\mathbf{S}^r \mathbf{v}_i = \lambda_i^r \mathbf{v}_i$, leading to the product $\lambda_i^{k-r-1}\lambda_i^r = \lambda_i^{k-1}$ being independent of r, so that

$$\sum_{i=1}^{N} \tilde{x}_i m_i \sum_{k=1}^{\infty} k h_k \lambda_i^{k-1} \mathbf{v}_i = \sum_{i=1}^{N} \tilde{x}_i m_i h'(\lambda_i) \mathbf{v}_i \tag{55}$$

where $h'(\lambda_i) = \sum_{k=1}^{\infty} k h_k \lambda_i^{k-1}$ is the derivative $h'(\lambda)$ of $h(\lambda)$ evaluated at $\lambda = \lambda_i$. In the case of (54) we note that

$$\sum_{i=1}^{N} \tilde{x}_i \mathbf{V} \sum_{k=0}^{\infty} h_k \sum_{r=0}^{k-1} \lambda_i^{k-r-1} \mathbf{\Lambda}^r \mathbf{V}^{\mathsf{H}} \mathbf{E}_U \mathbf{v}_i = \sum_{i=1}^{N} \tilde{x}_i \mathbf{V} \mathrm{diag}(\mathbf{g}_i) \mathbf{V}^{\mathsf{H}} \mathbf{E}_U \mathbf{v}_i \tag{56}$$

where $\mathbf{g}_i \in \mathbb{R}^N$ is such that

$$[\mathbf{g}_i]_j = \sum_{k=0}^{\infty} h_k \sum_{r=0}^{k-1} \lambda_i^{k-r-1} \lambda_j^r. \tag{57}$$

For $j = i$ we have $[\mathbf{g}_i]_i = h'(\lambda_i)$ while, for $j \neq i$, recall that $\sum_{r=0}^{k-1} \lambda_i^{k-r-1}\lambda_j^r = (\lambda_i^k - \lambda_j^k)/(\lambda_i - \lambda_j)$ so that

$$[\mathbf{g}_i]_j = \begin{cases} h'(\lambda_i) & \text{if } j = i \\ \frac{h(\lambda_i) - h(\lambda_j)}{\lambda_i - \lambda_j} & \text{if } j \neq i \end{cases}. \tag{58}$$

Note $\max_j |[\mathbf{g}_i]_j| \leq C$ due to hypothesis (13), $i = 1, \ldots, N$.

Using (55) and (56) back in (50), and computing the norm,

$$\left\| \left[\mathbf{H}(\hat{\mathbf{S}}) - \mathbf{H}(\mathbf{S})\right] \mathbf{x} \right\| \leq \|\mathbf{D}\tilde{\mathbf{x}}\| \tag{59}$$

$$+ \left\| \sum_{i=1}^{N} \tilde{x}_i m_i h'(\lambda_i) \mathbf{v}_i \right\| \tag{60}$$

$$+ \left\| \sum_{i=1}^{N} \tilde{x}_i \mathbf{V} \mathrm{diag}(\mathbf{g}_i) \mathbf{V}^{\mathsf{H}} \mathbf{E}_U \mathbf{v}_i \right\|. \tag{61}$$

For (60) we have

$$\left\| \sum_{i=1}^{N} \tilde{x}_i m_i h'(\lambda_i) \mathbf{v}_i \right\|^2 = \sum_{i=1}^{N} |\tilde{x}_i|^2 |m_i|^2 |h'(\lambda_i)|^2 \|\mathbf{v}_i\|^2 \tag{62}$$

since $\{\mathbf{v}_i\}$ conform an orthonormal basis. Then, we recall that $\|\mathbf{v}_i\|^2 = 1$ and, from hypothesis (14) we have $|m_i| \leq \varepsilon$ and from hypothesis (13), $|h'(\lambda_i)| \leq C$, so that

$$\left\| \sum_{i=1}^{N} \tilde{x}_i m_i h'(\lambda_i) \mathbf{v}_i \right\|^2 \leq \varepsilon^2 C^2 \sum_{i=1}^{N} |\tilde{x}_i|^2. \tag{63}$$

Recalling that $\sum_{i=1}^{N} |\tilde{x}_i|^2 = \|\tilde{\mathbf{x}}\|^2 = \|\mathbf{x}\|^2$ and applying square root, we finally bound (60) by

$$\left\| \sum_{i=1}^{N} \tilde{x}_i m_i h'(\lambda_i) \mathbf{v}_i \right\| \leq \varepsilon C \|\mathbf{x}\|. \tag{64}$$

Now, moving on to (61) and using triangle inequality together with submultiplicativity of the operator norm, we have

$$\left\| \sum_{i=1}^{N} \tilde{x}_i \mathbf{V} \mathrm{diag}(\mathbf{g}_i) \mathbf{V}^{\mathsf{H}} \mathbf{E}_U \mathbf{v}_i \right\| \leq \sum_{i=1}^{N} |\tilde{x}_i| \|\mathbf{V} \mathrm{diag}(\mathbf{g}_i) \mathbf{V}^{\mathsf{H}}\| \|\mathbf{E}_U\| \|\mathbf{v}_i\|. \tag{65}$$

We have $\|\mathbf{V}\mathrm{diag}(\mathbf{g}_i)\mathbf{V}^{\mathsf{H}}\| \leq C$ for all $i = 1, \ldots, N$ from (58) in combination with hypothesis (13), and also $\|\mathbf{v}_i\| = 1$. As for $\|\mathbf{E}_U\|$, we know from Lemma 1 that $\|\mathbf{E}_U\| \leq \varepsilon\delta$. Then,

$$\left\| \sum_{i=1}^{N} \mathbf{V} \mathrm{diag}(\mathbf{g}_i) \mathbf{V}^{\mathsf{H}} \mathbf{E}_U (\tilde{x}_i \mathbf{v}_i) \right\| \leq C \varepsilon \delta \sqrt{N} \|\mathbf{x}\| \tag{66}$$

where we used that $\sum_{i=1}^{N} |\tilde{x}_i| = \|\tilde{\mathbf{x}}_i\|_1 \leq \sqrt{N}\|\tilde{\mathbf{x}}\| = \sqrt{N}\|\mathbf{x}\|$.

Finally, for the second order term (59) stemming from the expansion of $\hat{\mathbf{S}}^k$, we obtain

$$\|\mathbf{D}\tilde{\mathbf{x}}\| \leq \mathcal{O}(\|\mathbf{E}\|^2)\|\mathbf{x}\|_2 \leq \mathcal{O}(\varepsilon^2)\|\mathbf{x}\|_2. \tag{67}$$

Using bound (64) in (60) and bound (66) in (61), together with the bound (67) we just obtained for (59), we obtain

$$\left\|[\mathbf{H}(\hat{\mathbf{S}})-\mathbf{H}(\mathbf{s})]\mathbf{x}\right\| \leq \varepsilon C\|\mathbf{x}\| + \varepsilon C\delta\sqrt{N}\|\mathbf{x}\| + \mathcal{O}(\varepsilon^2)\|\mathbf{x}\|.$$

We complete the proof by using that $\|\mathbf{x}\| = 1$ as per Def. 1 and recalling that we have assumed that $\mathbf{I}$ is the permutation that achieves the minimum norm of all $\mathbf{P} \in \mathcal{P}$. ■

APPENDIX C
STABILITY UNDER RELATIVE PERTURBATIONS

Proof of Theorem 2: Following from the fact that graph filters are permutation equivariant (Prop. 1), we can assume, without loss of generality, that $\mathbf{P}_0 = \mathbf{I}$ solves (18). From a first order expansion analogous to (48), where we use $\mathbf{ES} + \mathbf{SE}$ instead of just $\mathbf{E}$ as the second term, we obtain

$$\begin{aligned}&\mathbf{H}(\hat{\mathbf{S}}) - \mathbf{H}(\mathbf{S})\\ &\quad= \sum_{k=0}^{\infty} h_k \sum_{r=0}^{k-1} \left(\mathbf{S}^r\mathbf{E}\mathbf{S}^{k-r} + \mathbf{S}^{r+1}\mathbf{E}\mathbf{S}^{k-r-1}\right) + \mathbf{D}\end{aligned} \tag{68}$$

with $\mathbf{D}$ such that $\|\mathbf{D}\| = \mathcal{O}(\|\mathbf{E}\|^2)$, in analogy to (49).

Next, we consider the difference in the effects of the filter on an arbitrary graph signal $\mathbf{x}$ with finite energy $\|\mathbf{x}\| < \infty$ that has a GFT given by $\tilde{\mathbf{x}} = [\tilde{x}_1, \ldots, \tilde{x}_N]^{\mathsf{T}}$ so that $\mathbf{x} = \sum_{i=1}^N \tilde{x}_i \mathbf{v}_i$ for $\{\mathbf{v}_i\}_{i=1}^N$ the eigenvector basis of the GSO $\mathbf{S}$. Then,

$$\begin{aligned}\left[\mathbf{H}(\hat{\mathbf{S}}) - \mathbf{H}(\mathbf{S})\right]\mathbf{x} &= \sum_{i=1}^{N} \tilde{x}_i \mathbf{D}\mathbf{v}_i\\ &+ \sum_{i=1}^{N} \tilde{x}_i \sum_{k=0}^{\infty} h_k \sum_{r=0}^{k-1} \left(\mathbf{S}^r\mathbf{E}\mathbf{S}^{k-r} + \mathbf{S}^{r+1}\mathbf{E}\mathbf{S}^{k-r-1}\right)\mathbf{v}_i.\end{aligned} \tag{69}$$

Let us consider first the product $\mathbf{S}^{r+1}\mathbf{E}\mathbf{S}^{k-r-1}\mathbf{v}_i$ in (69). It is immediate that $\mathbf{S}^{k-r-1}\mathbf{v}_i = \lambda_i^{k-r-1}\mathbf{v}_i$, and, in combination with Lemma 1, we get

$$\begin{aligned}\mathbf{S}^{r+1}\mathbf{E}\mathbf{S}^{k-r-1}\mathbf{v}_i &= \lambda_i^{k-r-1}\mathbf{S}^{r+1}\left(m_i\mathbf{v}_i + \mathbf{E}_U\mathbf{v}_i\right)\\ &= m_i\lambda_i^k\mathbf{v}_i + \lambda_i^{k-r-1}\mathbf{S}^{r+1}\mathbf{E}_U\mathbf{v}_i.\end{aligned} \tag{70}$$

Analogously, for the second product, we get $\mathbf{S}^r\mathbf{E}\mathbf{S}^{k-r}\mathbf{v}_i = m_i\lambda_i^k\mathbf{v}_i + \lambda_i^{k-r}\mathbf{S}^r\mathbf{E}_U\mathbf{v}_i$. Then, using these results, we can write

$$\begin{aligned}&\sum_{i=1}^{N}\tilde{x}_i\sum_{k=0}^{\infty}h_k\sum_{r=0}^{k-1}(\mathbf{S}^r\mathbf{E}\mathbf{S}^{k-r} + \mathbf{S}^{r+1}\mathbf{E}\mathbf{S}^{k-r-1})\mathbf{v}_i\\ &= 2\sum_{i=1}^{N}\tilde{x}_i m_i\lambda_i h'(\lambda_i)\mathbf{v}_i + \sum_{i=1}^{N}\tilde{x}_i\mathbf{V}\mathrm{diag}(\mathbf{g}_i)\mathbf{V}^{\mathsf{H}}\mathbf{E}_U\mathbf{v}_i\end{aligned} \tag{71}$$

where, for the first term, we gathered the two equal terms $m_i\lambda_i^k\mathbf{v}_i$ and used the fact that $\sum_{k=0}^{\infty} h_k\lambda_i^k = \lambda_i h'(\lambda_i)$; and for the second term, we defined $\mathbf{g}_i \in \mathbb{R}^N$ as

$$\begin{aligned}[\mathbf{g}_i]_j &= \sum_{k=0}^{\infty} h_k \sum_{r=0}^{k-1}\left(\lambda_i^{k-r-1}[\mathbf{\Lambda}^{r+1}]_j + \lambda_i^{k-r}[\mathbf{\Lambda}^r]_j\right)\\ &= \begin{cases}\lambda_i h'(\lambda_i) & \text{if } i = j\\ \frac{\lambda_i+\lambda_j}{\lambda_i-\lambda_j}\left(h(\lambda_i) - h(\lambda_j)\right) & \text{if } i \neq j\end{cases}.\end{aligned} \tag{72}$$

Finally, we proceed to bound $\|(\mathbf{H}(\hat{\mathbf{S}}) - \mathbf{H}(\mathbf{S}))\mathbf{x}\|$. For the first term in (69), we simply have $\|\mathbf{D}\mathbf{x}\| \leq \mathcal{O}(\varepsilon^2)\|\mathbf{x}\|$ by definition of operator norm and the error of the first order approximation (68). For the second term in (69) we need to bound the two terms in (71). The first of the terms in (71) is bounded analogously to (64), noting that, in this case, $|m_i| \leq \varepsilon$ by means of (21), and $|\lambda_i h'(\lambda_i)| \leq C$ due to (19). For the second term in (71), we proceed analogously to (66), where now $\|\mathbf{E}_U\| \leq \varepsilon\delta$ and $\|\mathbf{V}\mathrm{diag}(\mathbf{g}_i)\mathbf{V}^{\mathsf{H}}\| \leq 2\,C$, following the condition imposed by integral Lipschitz filters (19). All of these results together yield

$$\left\|(\mathbf{H}(\hat{\mathbf{S}}) - \mathbf{H}(\mathbf{S}))\mathbf{x}\right\| \leq 2\,C\varepsilon\|\mathbf{x}\| + 2\,C\varepsilon\delta\sqrt{N}\|\mathbf{x}\| + \mathcal{O}(\varepsilon^2)\|\mathbf{x}\|.$$

We complete the proof by using that $\|\mathbf{x}\| = 1$ as per Def. 1, and recalling that we have assumed that $\mathbf{I}$ is the permutation that achieves the minimum norm of all $\mathbf{P} \in \mathcal{P}$. ■

Proof of Theorem 3: The proof is analogous to that of Theorem 2, with the following main difference. Denote by m_i, $i = 1, \ldots, N$, the eigenvalues of $\mathbf{E} = \mathbf{U}\mathbf{M}\mathbf{U}^{\mathsf{H}}$. If we order these eigenvalues as $|m_1| \leq \cdots \leq |m_N|$, we know that $\|\mathbf{E}\| = |m_N|$ and condition (23) becomes equivalent to $\|\mathbf{E}/m_N - \mathbf{I}\| \leq \varepsilon$. This can be used to write $\mathbf{E}\mathbf{v}_i$, not as in Lemma 1, but as

$$\mathbf{E}\mathbf{v}_i = \sum_{n=1}^{N} m_n\mathbf{u}_n\mathbf{u}_n^{\mathsf{H}}\mathbf{v}_i = m_N\sum_{n=1}^{N}(1+\delta_n)\mathbf{u}_n\mathbf{u}_n^{\mathsf{H}}\mathbf{v}_i. \tag{73}$$

where $m_n/m_N = 1 + \delta_n$ for all $n = 1, \ldots, N$ with $|\delta_n| \leq \varepsilon$ in virtue of (23), yielding

$$\mathbf{E}\mathbf{v}_i = m_N\mathbf{v}_i + m_N\mathbf{w}_i, \quad \mathbf{w}_i = \sum_{n=1}^{N}\delta_n\mathbf{u}_n\mathbf{u}_n^{\mathsf{H}}\mathbf{v}_i. \tag{74}$$

Using this expression in (71), it becomes

$$\begin{aligned}&\sum_{i=1}^{N}\tilde{x}_i\sum_{k=0}^{\infty}h_k\sum_{r=0}^{k-1}(\mathbf{S}^r\mathbf{E}\mathbf{S}^{k-r} + \mathbf{S}^{r+1}\mathbf{E}\mathbf{S}^{k-r-1})\mathbf{v}_i\\ &= 2m_N\sum_{i=1}^{N}\tilde{x}_i\lambda_i h'(\lambda_i)\mathbf{v}_i + m_N\sum_{i=1}^{N}\tilde{x}_i\mathbf{V}\mathrm{diag}(\mathbf{g}_i)\mathbf{V}^{\mathsf{H}}\mathbf{w}_i.\end{aligned} \tag{75}$$

Noting that $|m_N| \leq \varepsilon$ because (21) holds, and that

$$\|\mathbf{w}_i\| \leq \left\|\sum_{n=1}^{N}\delta_n\mathbf{u}_n\mathbf{u}_n^{\mathsf{H}}\right\|\|\mathbf{v}_i\| = \max_{n=1,\ldots,N}|\delta_n| \leq \varepsilon \tag{76}$$

we observe that the first term of (75) is bounded above by $2\,C\varepsilon$ while the second term is bounded by $2\varepsilon^2 C\sqrt{N} = \mathcal{O}(\varepsilon^2)$, completing the proof. ■

APPENDIX D
PERMUTATION EQUIVARIANCE OF GNNS

Proof of Prop. 2: First, we obtain the output $\hat{\mathbf{z}}_\ell^{fg}$ of (25) when the input is $\hat{\mathbf{x}}_{\ell-1}^g = \mathbf{P}^\mathsf{T}\mathbf{x}_{\ell-1}^g$, operating on the correspondingly permuted GSO $\hat{\mathbf{S}} = \mathbf{P}^\mathsf{T}\mathbf{S}\mathbf{P}$. Since we know that application of graph filters is permutation equivariant (Prop. 1), we have that the output $\hat{\mathbf{z}}_\ell^{fg}$ is

$$\hat{\mathbf{z}}_\ell^{fg} = \mathbf{H}_\ell^{fg}(\hat{\mathbf{S}})\hat{\mathbf{x}}_{\ell-1}^g = \mathbf{P}^\mathsf{T}\mathbf{H}_\ell^{fg}(\mathbf{S})\mathbf{x}_{\ell-1}^g = \mathbf{P}^\mathsf{T}\mathbf{z}_\ell^{fg}. \tag{77}$$

Next, we note that, for any pointwise function σ applied to each element of a vector, it holds that $\sigma(\hat{\mathbf{x}}) = \sigma(\mathbf{P}^\mathsf{T}\mathbf{x}) = \mathbf{P}^\mathsf{T}\sigma(\mathbf{x})$. Therefore, it is immediate that

$$\hat{\mathbf{x}}_\ell^f = \sigma\left[\sum_g \hat{\mathbf{z}}_\ell^{fg}\right] = \sigma\left[\sum_g \mathbf{P}^\mathsf{T}\mathbf{z}_\ell^{fg}\right] = \mathbf{P}^\mathsf{T}\sigma\left[\sum_g \mathbf{z}_\ell^{fg}\right] \tag{78}$$

where we note that the last equality is $\mathbf{P}^\mathsf{T}\mathbf{x}_\ell^f$ using (26). Since (77)–(78) hold, we have that each layer ℓ of the GNN is permutation equivariant. And noting that this holds for any $\ell = 1, \ldots, L$ completes the proof. ■

APPENDIX E
GRAPH NEURAL NETWORKS STABILITY

Proof of Theorem 4: Let us consider the general case where we have F_ℓ features per layer, for $\ell = 0, \ldots, L$, where F_0 are the number of input features and F_L the number of output features. That is, the input to the GNN is the collection of F_0 graph signals $\mathbf{x} = \{\mathbf{x}^g\}_{g=1}^{F_0}$ and the output $\Phi(\mathbf{S}, \mathbf{x})$ is a collection of F_L graph signals $\{\mathbf{x}_L^f\}_{f=1}^{F_L}$ [cf. (29)]. In this context, we are interested in the difference between the output of the GNNs when evaluated on different shift operators $\mathbf{S}$ and $\hat{\mathbf{S}}$

$$\|\Phi(\mathbf{S}, \mathbf{x}) - \Phi(\hat{\mathbf{S}}, \mathbf{x})\|^2 = \sum_{f=1}^{F_L} \|\mathbf{x}_L^f - \hat{\mathbf{x}}_L^f\|^2. \tag{79}$$

We use $\hat{\cdot}$ to denote an operation acting on $\hat{\mathbf{S}}$ instead of $\mathbf{S}$. For example, we denote $\mathbf{H}_\ell^{fg}(\mathbf{S}) = \mathbf{H}_\ell^{fg}$ and $\mathbf{H}_\ell^{fg}(\hat{\mathbf{S}}) = \hat{\mathbf{H}}_\ell^{fg}$ for two filters with the same coefficients $\mathbf{h}_\ell^{fg}$ but acting on different shift operators $\mathbf{S}$ and $\hat{\mathbf{S}}$ [cf. (2)–(3)]. Now, focusing on one of the features of the last layer [cf. (25), (26), (29)]

$$\|\mathbf{x}_L^f - \hat{\mathbf{x}}_L^f\| = \left\|\sigma\left(\sum_{g=1}^{F_{L-1}} \mathbf{H}_L^{fg}\mathbf{x}_{L-1}^g\right) - \sigma\left(\sum_{g=1}^{F_{L-1}} \hat{\mathbf{H}}_L^{fg}\hat{\mathbf{x}}_{L-1}^g\right)\right\| \tag{80}$$

and applying Lipschitz continuity of the nonlinearity (28) by which $|\sigma(b) - \sigma(a)| \leq C_\sigma|b - a|$ with $C_\sigma = 1$, followed by the triangular inequality, we get

$$\|\mathbf{x}_L^f - \hat{\mathbf{x}}_L^f\| \leq C_\sigma \sum_{g=1}^{F_{L-1}} \left\|\mathbf{H}_L^{fg}\mathbf{x}_{L-1}^g - \hat{\mathbf{H}}_L^{fg}\hat{\mathbf{x}}_{L-1}^g\right\|. \tag{81}$$

Adding and subtracting $\hat{\mathbf{H}}_L^{fg}\mathbf{x}_{L-1}^g$ from the terms in the sum, and using the triangular inequality once more, we get

$$\left\|\mathbf{H}_L^{fg}\mathbf{x}_{L-1}^g - \hat{\mathbf{H}}_L^{fg}\hat{\mathbf{x}}_{L-1}^g\right\| \leq \left\|\left(\mathbf{H}_L^{fg} - \hat{\mathbf{H}}_L^{fg}\right)\mathbf{x}_{L-1}^g\right\| + \left\|\hat{\mathbf{H}}_L^{fg}\left(\mathbf{x}_{L-1}^g - \hat{\mathbf{x}}_{L-1}^g\right)\right\|. \tag{82}$$

The definition of operator norm, implies that

$$\left\|\mathbf{H}_L^{fg}\mathbf{x}_{L-1}^g - \hat{\mathbf{H}}_L^{fg}\hat{\mathbf{x}}_{L-1}^g\right\| \leq \left\|\mathbf{H}_L^{fg} - \hat{\mathbf{H}}_L^{fg}\right\|\left\|\mathbf{x}_{L-1}^g\right\| + \left\|\hat{\mathbf{H}}_L^{fg}\right\|\left\|\mathbf{x}_{L-1}^g - \hat{\mathbf{x}}_{L-1}^g\right\|. \tag{83}$$

For the first term in the inequality (83) we can use the hypothesis that $\|\mathbf{H}_l^{fg} - \hat{\mathbf{H}}_l^{fg}\| \leq \Delta\varepsilon$ for all layers $l = 1, \ldots, L$, while for the second term, we can use that $\|\hat{\mathbf{H}}_l^{fg}\| \leq B = 1$ for all layers. Using these two facts in (83) and substituting back in (81),

$$\|\mathbf{x}_L^f - \hat{\mathbf{x}}_L^f\|_2 \leq C_\sigma \sum_{g=1}^{F_{L-1}} \left(\Delta\varepsilon\|\mathbf{x}_{L-1}^g\| + B\|\mathbf{x}_{L-1}^g - \hat{\mathbf{x}}_{L-1}^g\|\right). \tag{84}$$

We observe that (84) shows a recursion, where the bound at layer L depends on the bound at layer $L-1$ as well as the norm of the features at layer $L-1$, summed over all features. That is, for an arbitrary layer $\ell \in \{1, \ldots, L\}$, we have

$$\|\mathbf{x}_\ell^f - \hat{\mathbf{x}}_\ell^f\| \leq C_\sigma \sum_{g=1}^{F_{\ell-1}} \left(\Delta\varepsilon\|\mathbf{x}_{\ell-1}^g\| + B\|\mathbf{x}_{\ell-1}^g - \hat{\mathbf{x}}_{\ell-1}^g\|\right). \tag{85}$$

with initial conditions given by the input features $\mathbf{x}_0^g = \mathbf{x}^g$ for $g = 1, \ldots, F_0$, so that $\|\mathbf{x}_0^g - \hat{\mathbf{x}}_0^g\| = \|\mathbf{x}^g - \mathbf{x}^g\| = 0$. For the first step to solve the recursion (85), we compute the norm $\|\mathbf{x}_\ell^f\|$. We observe that

$$\|\mathbf{x}_\ell^f\| \leq C_\sigma \left\|\sum_{g=1}^{F_{l-1}} \mathbf{H}_\ell^{fg}\mathbf{x}_{\ell-1}^g\right\| \leq C_\sigma B \sum_{g=1}^{F_{l-1}} \|\mathbf{x}_{\ell-1}^g\| \tag{86}$$

where we used the triangle inequality, followed by the bound on the filters. Solving (86) with initial condition $\|\mathbf{x}_0^g\| = \|\mathbf{x}^g\|$,

$$\|\mathbf{x}_\ell^f\| \leq (C_\sigma B)^\ell \prod_{\ell'=1}^{\ell-1} F_{\ell'} \sum_{g=1}^{F_0} \|\mathbf{x}^g\|. \tag{87}$$

Using (87) back in recursion (85) and solving it with the corresponding initial conditions, we get

$$\left\|\mathbf{x}_\ell^f - \hat{\mathbf{x}}_\ell^f\right\| \leq \Delta\varepsilon(C_\sigma B)^{\ell-1}\left(\sum_{\ell'=1}^{\ell} C_\sigma^{\ell'}\right)\left(\prod_{\ell'=1}^{\ell-1} F_{\ell'}\right)\sum_{g=1}^{F_0}\|\mathbf{x}^g\|.$$

Evaluating this for $\ell = L$ and using it back in (84), (79) yields

$$\left\|\Phi(\mathbf{S}, \mathbf{x}) - \Phi(\hat{\mathbf{S}}, \mathbf{x})\right\|^2 = \sum_{f=1}^{F_L}\left\|\mathbf{x}_L^f - \hat{\mathbf{x}}_L^f\right\|^2 \leq \sum_{f=1}^{F_L}\left(\Delta\varepsilon(C_\sigma B)^{L-1}\sum_{\ell=1}^{L} C_\sigma^\ell \prod_{\ell=1}^{L-1} F_\ell \sum_{g=1}^{F_0}\|\mathbf{x}^g\|\right)^2. \tag{88}$$

Noting that no term in the sum of (88) depends on f, and subsequently applying a square root, we get

$$\left\|\Phi(\mathbf{S}, \mathbf{x}) - \Phi(\hat{\mathbf{S}}, \mathbf{x})\right\| \leq \sqrt{F_L}\Delta\varepsilon(C_\sigma B)^{L-1}\sum_{\ell=1}^{L} C_\sigma^\ell \prod_{\ell=1}^{L-1} F_\ell \sum_{g=1}^{F_0}\|\mathbf{x}^g\|. \tag{89}$$

Finally, setting $F_L = F_0 = 1$ yields $\sqrt{F_L} = 1$ and $\sum_{g=1}^{F_0} \|\mathbf{x}^g\| = \|\mathbf{x}\|$, setting $F_1 = \cdots = F_{L-1} = F$, $B = 1$ and $C_\sigma = 1$ yields $B^{L-1} = 1$ and $\sum_{\ell=1}^{L} C_\sigma^\ell = L$, respectively. This completes the proof. ■

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 85–117, 2015.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ser. The Adaptive Computation and Machine Learning Series. Cambridge, MA: The MIT Press, 2016.

[3] S. Mallat, "Group invariant scattering," *Commun. Pure, Appl. Math.*, vol. 65, no. 10, pp. 1331–1398, Oct. 2012.

[4] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, Aug. 2013.

[5] D. Yin, R. G. Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer, "A Fourier perspective on model robustness in computer vision," in *Proc. 33rd Conf. Neural Inf. Process. Syst.*, Vancouver, BC: Neural Inform. Process. Syst. Found., Dec. 2019, pp. 13 276–13 286.

[6] H. Wang, X. Wu, Z. Huang, and E. P. Xing, "High-frequency component helps explain the generalization of convolutional neural networks," in *Proc. Conf. Comput. Vis. Pattern Recognit*. Comput. Vision Found., 14–19 Jun. 2020, pp. 8684–8694.

[7] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[8] A. Sandyhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, Jun. 2014.

[9] S. Segarra, A. G. Marques, and A. Ribeiro, "Optimal graph-filter design and applications to distributed linear network operators," *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 4117–4131, Aug. 2017.

[10] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and deep locally connected networks on graphs," in *Proc. 2nd Int. Conf. Learn. Representations*, Banff, AB, 14-16 Apr. 2014, pp. 1–14.

[11] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. 30th Conf. Neural Inform. Process. Syst.* Barcelona, Spain: Neural Inform. Process. Found., Dec. 2016, pp. 3844–3858.

[12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Represent.*, Toulon, France, Apr. 2017, pp. 24–26.

[13] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 4, pp. 1034–1049, Feb. 2019.

[14] R. Ying, R. He, K. Chen, P. Eksombatchai, W. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery & Data Mining*. London, U.K.: Assoc. Comput. Mach., Aug. 2018, pp. 974–983.

[15] D. Owerko, F. Gama, and A. Ribeiro, "Predicting power outages using graph neural networks," in *Proc. IEEE Global Conf. Signal Inf. Process.*, Anaheim, CA: IEEE, Nov. 2018, pp. 743–747.

[16] D. Owerko, F. Gama, and A. Ribeiro, "Optimal power flow using graph neural networks," in *Proc. 45th IEEE Int. Conf. Acoust., Speech, Signal Process.*, Barcelona, Spain: IEEE, 4–8 May 2020.

[17] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," in *Proc. Conf. Robot Learn.*, vol. 100. Osaka, Japan: Proc. Mach. Learning Res., 30 Oct.-1 Nov. 2019, pp. 671–682.

[18] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *Proc. 19th Int. Conf. Auton. Agents Multi-Agent Syst.*, Auckland, New Zealand: IFAAMAS, 9–13 May 2020.

[19] D. Zou and G. Lerman, "Graph convolutional neural networks via scattering," *Appl. Comput. Harmonic Anal.*, vol. 49, no. 3, pp. 1046–1074, Nov. 2020.

[20] F. Gama, A. Ribeiro, and J. Bruna, "Diffusion scattering transforms on graphs," in *Proc. 7th Int. Conf. Learn. Representations*, New Orleans, LA, 6-9 May 2019, pp. 1–12.

[21] R. R. Coifman and S. Lafon, "Diffusion maps," *Appl. Comput. Harmonic Anal.*, vol. 21, no. 1, pp. 5–30, Jul. 2006.

[22] R. Levie, E. Isufi, and G. Kutyniok, "On the transferability of spectral graph filters," in *Proc. 13th Int. Conf. Sampling Theory Appl.*, Bordeaux, France: IEEE, 8–12 Jul. 2019, pp. 1–5.

[23] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Proc. 7th Int. Conf. Learn. Representations*, New Orleans, LA, 6–9 May 2019, pp. 1–17.

[24] H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman, "Invariant and equivariant graph networks," in *Proc. 7th Int. Conf. Learn. Representations*, New Orleans, LA, 6–9 May 2019, pp. 1–14.

[25] N. Keriven and G. Peyré, "Universal invariant and equivariant graph neural networks," in *Proc. 33rd Conf. Neural Inform. Process. Syst.* Vancouver, BC: Neural Inform. Process. Syst. Found., Dec. 2019, pp. 7092–7101.

[26] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery & Data Mining*., London, U.K.: Assoc. Comput. Mach., Aug. 2018, pp. 2847–2856.

[27] J. Dai, J. Li, T. Tian, X. Huang, X. Wang, J. Zu, and L. Song, "Adversarial attack on graph structured data," in *Proc. 35th Int. Conf. Mach. Learn.*, Stockholm, Sweden: Proc. Mach. Learning Res., 10–15 Jul. 2018, pp. 1115–1124.

[28] A. Bojchevski and S. Günnemann, "Certifiable robustness to graph perturbations," in *Proc. 33rd Conf. Neural Inform. Process. Syst.*, Vancouver, BC: Neural Inform. Process. Syst. Found., 8–14 Dec. 2019, pp. 8319–8330.

[29] D. Zügner and S. Günnemann, "Certifiable robustness and robust training for graph convolutional networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery & Data Mining*., Anchorage, AK: Assoc. Comput. Mach., Aug. 2019, pp. 246–256.

[30] I. Daubechies, *Ten Lectures Wavelets*, ser. CBMS-NSF Regional Conf. Series Appl. Math. Philadelphia, PA: SIAM, 1992, vol. 61.

[31] R. R. Coifman and M. Maggioni, "Diffusion wavelets," *Appl. Comput. Harmonic Anal.*, vol. 21, no. 1, pp. 53–94, Jul. 2006.

[32] D. I. Shuman, C. Wiesmeyr, N. Holighaus, and P. Vandergheynst, "Spectrum-adapted tight graph wavelet and vertex-frequency frames," *IEEE Trans. Signal Process.*, vol. 63, no. 16, pp. 4223–4235, Aug. 2015.

[33] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmonic Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2011.

[34] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional graph neural networks," in *Proc. 53rd Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, CA: IEEE, Nov. 2019.

[35] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "From graph filters to graph neural networks," 8 Aug. 2020. [Online]. Available: https://arxiv.org/abs/2003.03777

[36] E. Isufi, F. Gama, and A. Ribeiro, "EdgeNets: Edge varying graph neural networks," 12 Mar. 2020. [Online]. Available: https://arxiv.org/abs/2001.07620

[37] W. Huang, A. G. Marques, and A. Ribeiro, "Rating prediction via graph signal processing," *IEEE Trans. Signal Process.*, vol. 66, no. 19, pp. 5066–5081, Oct. 2018.

[38] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interactive Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Jan. 2016.

[39] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Trans. Signal, Inform. Process. Netw.*, vol. 3, no. 3, pp. 467–483, Sep. 2017.