2022 IEEE Data Science and Learning Workshop (DSLW) | 978-1-6654-5426-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/DSLW53931.2022.9820575

NODE-VARIANT GRAPH FILTERS IN GRAPH NEURAL NETWORKS

Fernando Gama, Brendon G. Anderson†, Somayeh Sojoudi†*

∗ Department of Electrical and Computer Engineering, Rice University, Houston, TX
† Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA

ABSTRACT

Graph neural networks (GNNs) have been successfully employed in a myriad of applications involving graph signals. Theoretical findings establish that GNNs use nonlinear activation functions to create low-eigenvalue frequency content that can be processed in a stable manner by subsequent graph convolutional filters. However, the exact shape of the frequency content created by nonlinear functions is not known and cannot be learned. In this work, we use node-variant graph filters (NVGFs) –which are linear filters capable of creating frequencies– as a means of investigating the role that frequency creation plays in GNNs. We show that, by replacing nonlinear activation functions by NVGFs, frequency creation mechanisms can be designed or learned. By doing so, the role of frequency creation is separated from the nonlinear nature of traditional GNNs. Simulations on graph signal processing problems are carried out to pinpoint the role of frequency creation.

1. INTRODUCTION

Graph neural networks (GNNs) [1, 2] are learning architectures that have been successfully applied to a wide array of graph signal processing (GSP) problems ranging from recommendation systems [3, 4] and authorship attribution [5, 6], to physical network problems including wireless communications [7], control [8], and sensor networks [9]. In the context of GSP, frequency analysis has been successful at providing theoretical insight into the observed success of GNNs [10–12] and graph scattering transforms [13, 14].

Of particular interest is the seminal work by Mallat [15,16], concerning discrete-time signals and images, that argues that the improved performance of convolutional neural networks (CNNs) over linear convolutional filters is due to the activation functions. Concretely, nonlinear activation functions allow for high-frequency content to be spread into lower frequencies, where it can be processed in a stable manner—a feat that cannot be achieved by convolutional filters alone.Leveraging the notion of graph Fourier transform (GFT) [17], these results have been extended to GNNs [10], establishing that the use of functions capable of creating low-eigenvalue frequency content allows them to be robust to changes in the graph topology, facilitating scalability and transferability [18].

While nonlinear activation functions play a key role in the creation of low-eigenvalue frequency content, it is not possible to know the exact shape in which this frequency content is actually generated. Node-variant graph filters (NVGFs) [19], which essentially assign a different filter tap to each node in the graph, are also able to generate frequency content. Different from nonlinear activation functions, this frequency content can be exactly computed, given the filter taps. Thus, by learning or carefully designing these filter taps, it is possible to know exactly how the frequency content is being generated.

The NVGF is a linear filter, which means that replacing the nonlinear activation functions with NVGFs actually renders the whole architecture a linear one. Therefore, by comparing the performance of this architecture to that of a traditional GNN, it is possible to isolate the role of frequency creation in the overall performance of the architecture, from that of the nonlinear nature of mappings. The contributions of this paper can be summarized as follows:

1. We introduce NVGFs as a means of replacing nonlinear activation functions, motivated by their ability to create frequencies. We obtain closed-form expressions for the frequency response of NVGFs as a function of the filter taps.
2. We prove that NVGFs are Lipschitz continuous with respect to changes in the underlying graph topology.
3. We put forth a framework for designing NVGFs.
4. We propose a GNN architecture where the nonlinear activation function is replaced by a NVGF. The filter taps of the NVGF can be either learned or designed. The resulting architecture decouples the role of frequency creation from the nonlinear nature of the GNN.
5. We investigate the problem of authorship attribution to demonstrate, both quantitatively and qualitatively, the role of frequency creation in the performance of a GNN, and its relationship to the nonlinear nature of traditional architectures.

In essence, we show that nonlinear activation functions are not strictly required for creating frequencies, as originally thought in [10, 15], but that linear NVGF activation functions are sufficient. Furthermore, we demonstrate that this frequency content can be learned with respect to the specific problem under study. All proofs, as well as the code and further simulations, can be found online[1]

Related work. GNNs constitute a very active area of research [20, 21]. From a GSP perspective, spectral filtering is used in [22], it is then replaced by computationally simpler Chebyshev polynomials [23], and subsequently by general graph convolutional filters [6]. GNNs were also adopted in non-GSP problems [24–26]. The proposed replacement of nonlinear activation functions by NVGFs creates a linear architecture that uses both convolutional and non-convolutional graph filters.

NVGFs are first introduced in [19] to extend time-variant filters into the realm of graph signals. In that work, NVGFs are used to optimally approximate linear operators in a distributed manner. In this paper, we focus on the frequency response of NVGFs and on their capacity to create frequency content.

Leveraging the notion of GFT, [10] shows that a GNN is Lipschitz continuous to small changes in the underlying graph structure. Likewise, frequency analysis has been used to understand graph scattering transforms, where the filters used in the GNN are chosen to be wavelets (and not learned) [13, 14]. In this paper, we focus on the role of frequency creation that is put forth in [10, 15], and study NVGFs as linear mechanisms for achieving this.

This work is partially supported by NSF and ONR.

[1]Proofs: `http://arxiv.org/abs/2106.00089`
Code: `http://github.com/fgfgama/nvgf`

2. THE NODE-VARIANT GRAPH FILTER

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected, possibly weighted, graph with a set of N nodes $\mathcal{V} = \{v_1, \ldots, v_N\}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Define a graph signal as the function $\mathsf{x} : \mathcal{V} \to \mathbb{R}$ that associates a scalar value to each node. For a fixed ordering of the nodes, the graph signal can be conveniently described by means of a vector $\mathbf{x} \in \mathbb{R}^N$ such that the i^{th} entry $[\mathbf{x}]_i$ is the value x_i of the signal on node v_i, i.e., $[\mathbf{x}]_i = x_i = \mathsf{x}(v_i)$.

Describing a graph signal as the vector $\mathbf{x} \in \mathbb{R}^N$ is mathematically convenient but carries no information about the underlying graph topology that supports it. This information can be recovered by defining a graph matrix description (GMD) $\mathbf{S} \in \mathbb{R}^{N \times N}$ which is a matrix that respects the sparsity pattern of the graph, i.e., $[\mathbf{S}]_{ij} = 0$ for all distinct indices i and j such that $(v_j, v_i) \notin \mathcal{E}$. Examples of GMDs widely used include the adjacency matrix, the Laplacian matrix, and their corresponding normalizations [27–29].

The GMD $\mathbf{S}$ can thus be leveraged to process the graph signal $\mathbf{x}$ in such a way that the underlying graph structure is exploited. The most elementary example is the linear map $\mathsf{S} : \mathbb{R}^N \to \mathbb{R}^N$ between graph signals given by $\mathbf{y} = \mathsf{S}(\mathbf{x}) = \mathbf{S}\mathbf{x}$. This linear map is a linear combination of the information located in the one-hop neighborhood of each node:

$$y_i = [\mathbf{y}]_i = \sum_{j=1}^{N} [\mathbf{S}]_{ij} [\mathbf{x}]_j = \sum_{j: v_j \in \mathcal{N}_i \cup \{v_i\}} s_{ij} x_j \tag{1}$$

where $\mathcal{N}_i = \{v_j : (v_j, v_i) \in \mathcal{E}\}$ is the set of nodes that share an edge with node v_i. The last equality in the above equation is due to the sparsity pattern of the GMD $\mathbf{S}$.

More generally, a graph filter $\mathsf{H}(\cdot; \mathbf{S}) : \mathbb{R}^N \to \mathbb{R}^N$ is a mapping between graph signals that leverages the structure encoded in $\mathbf{S}$ [19]. In particular, linear shift-invariant graph filters (LSIGFs) are those that can be built as a polynomial in $\mathbf{S}$:

$$\mathsf{H}^{\text{lsi}}(\mathbf{x}; \mathbf{S}) = \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x} = \boldsymbol{H}^{\text{lsi}}(\mathbf{S})\mathbf{x} \tag{2}$$

where $\boldsymbol{H}^{\text{lsi}}(\mathbf{S}) = \sum_{k=0}^{K} h_k \mathbf{S}^k$ with $h_k \in \mathbb{R}$. Note that $\mathsf{H}^{\text{lsi}}(\mathbf{x}; \mathbf{S})$ is written in the form of $\boldsymbol{H}^{\text{lsi}}(\mathbf{S})\mathbf{x}$ to emphasize that the function is linear in the input $\mathbf{x}$, i.e., $\mathbf{x}$ is multiplied by a matrix $\boldsymbol{H}^{\text{lsi}}(\mathbf{S})$ that is parametrized by $\mathbf{S}$. LSIGFs inherit their name from the fact that they satisfy the property that $\boldsymbol{H}^{\text{lsi}}(\mathbf{S})\mathbf{S}\mathbf{x} = \mathbf{S}\boldsymbol{H}^{\text{lsi}}(\mathbf{S})\mathbf{x}$. The set of polynomial coefficients $\{h_k\}_{k=0}^{K}$ are called filter taps, and can be collected in a vector $\mathbf{h} \in \mathbb{R}^{K+1}$ defined as $[\mathbf{h}]_{k+1} = h_k$ for all $k \in \{0, \ldots, K\}$. Note that the term $\mathbf{S}^k\mathbf{x}$ is a convenient mathematical formulation, but in practice $\mathbf{S}^k\mathbf{x}$ is computed by exchanging information k times with one-hop neighbors, i.e., there are no matrix powers involved. In general, GSP regards $\mathbf{S}$ as given by the structure of the problem, and regards $\mathbf{x}$ as the actionable data [27–29].

This paper focuses on NVGFs [19], which are linear filters that assign a different filter tap to each node, for each application of $\mathbf{S}$. This can be compactly written as follows:

$$\mathsf{H}^{\text{nv}}(\mathbf{x}; \mathbf{S}) = \sum_{k=0}^{K} \mathsf{diag}(\mathbf{h}^{(k)}) \mathbf{S}^k \mathbf{x} = \boldsymbol{H}^{\text{nv}}(\mathbf{S})\mathbf{x} \tag{3}$$

where $\mathbf{h}^{(k)} \in \mathbb{R}^N$ and $\mathsf{diag}(\cdot)$ is the diagonal operator that takes a vector and creates a diagonal matrix with the elements of the vector in the diagonal. Since the NVGF is linear in the input, it holds that $\mathsf{H}^{\text{nv}}(\mathbf{x}; \mathbf{S}) = \boldsymbol{H}^{\text{nv}}(\mathbf{S})\mathbf{x}$, where $\boldsymbol{H}^{\text{nv}}(\mathbf{S}) = \sum_{k=0}^{K} \mathsf{diag}(\mathbf{h}^{(k)})\mathbf{S}^k$. The i^{th} entry of the vector, $[\mathbf{h}^{(k)}]_i = h_{ik}$, is the filter tap that node v_i uses to weigh the information incoming after k exchanges with its neighbors. The set of all filter taps can be conveniently collected in a matrix $\mathbf{H} \in \mathbb{R}^{N \times (K+1)}$ where the $(k+1)^{\text{th}}$ column is $\mathbf{h}^{(k)} \in \mathbb{R}^N$ and the i^{th} row, denoted by $\mathbf{h}_i \in \mathbb{R}^{K+1}$, contains the $K+1$ filter taps used by node v_i, i.e., $[\mathbf{h}_i]_{k+1} = h_{ik}$.

The LSIGF in (2) and the NVGF in (3) are both linear and local processing operators. They depend linearly on the input graph signal $\mathbf{x}$ as indicated by the matrix multiplication notation in (2) and in (3). They are local in the sense that, to compute the output, each node requires information relayed directly by their immediate neighbors. The LSIGF is characterized by the collection of $K+1$ filter taps. The NVGF, on the other hand, is characterized by $N(K+1)$ filter taps. It is noted that while the NVGF requires additional memory to store more parameters, this can be distributed throughout the graph. It is also observed that both the LSIGF and the NVGF have the same computational complexity.

3. FREQUENCY ANALYSIS

The GMD $\mathbf{S} \in \mathbb{R}^{N \times N}$ can be used to define a spectral representation of the graph signal $\mathbf{x} \in \mathbb{R}^N$ [17]. Since the graph is undirected, assume that $\mathbf{S}$ is symmetric so that it can be diagonalized by an orthonormal basis of eigenvectors $\{\mathbf{v}_i\}_{i=1}^{N}$, where $\mathbf{v}_i \in \mathbb{R}^N$ and $\mathbf{S}\mathbf{v}_i = \lambda_i \mathbf{v}_i$, with $\lambda_i \in \mathbb{R}$ being the corresponding eigenvalue. Then, it holds that $\mathbf{S} = \mathbf{V}\mathsf{diag}(\boldsymbol{\lambda})\mathbf{V}^{\mathsf{T}}$, where the i^{th} column of $\mathbf{V}$ is $\mathbf{v}_i$ and where $\boldsymbol{\lambda} \in \mathbb{R}^N$ is given by $[\boldsymbol{\lambda}]_i = \lambda_i$ for $i = 1, \ldots, N$. We assume throughout this paper that the eigenvalues are distinct, which is typically the case for random connected graphs.

The spectral representation of a graph signal $\mathbf{x}$ with respect to its underlying graph support described by $\mathbf{S}$ is given by

$$\tilde{\mathbf{x}} = \mathbf{V}^{\mathsf{T}}\mathbf{x} \tag{4}$$

where $\tilde{\mathbf{x}} \in \mathbb{R}^N$, see [17]. The spectral representation $\tilde{\mathbf{x}}$ of the graph signal $\mathbf{x}$ contains the coordinates of representing $\mathbf{x}$ on the eigenbasis $\{\mathbf{v}_i\}_{i=1}^{N}$ of the support matrix $\mathbf{S}$. The resulting vector $\tilde{\mathbf{x}}$ is known as the frequency response of the signal $\mathbf{x}$. The i^{th} entry $[\tilde{\mathbf{x}}]_i = \mathbf{v}_i^{\mathsf{T}}\mathbf{x} = \tilde{x}_i \in \mathbb{R}$ of the frequency response $\tilde{\mathbf{x}}$ measures how much the i^{th} eigenvector $\mathbf{v}_i$ contributes to the signal $\mathbf{x}$. The operation in (4) is called the GFT, and thus the frequency response $\tilde{\mathbf{x}}$ is often referred to as the GFT of the signal $\mathbf{x}$.

The GFT offers an alternative representation of the graph signal $\mathbf{x}$ that takes into account the graph structure in $\mathbf{S}$. The effect of a filter can be characterized in the spectral domain by computing the GFT of the output. For instance, when considering a LSIGF, the spectral representation of the output $\mathbf{y} = \boldsymbol{H}^{\text{lsi}}(\mathbf{S})\mathbf{x}$ is

$$\tilde{\mathbf{y}} = \mathbf{V}^{\mathsf{T}} \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x} = \sum_{k=0}^{K} h_k \mathsf{diag}(\boldsymbol{\lambda}^k)\tilde{\mathbf{x}} = \mathsf{diag}(\tilde{\mathbf{h}})\tilde{\mathbf{x}} \tag{5}$$

where the eigendecomposition of $\mathbf{S}$, the GFT of $\mathbf{x}$, and the fact that $\mathbf{S}^k = \mathbf{V}\mathsf{diag}(\boldsymbol{\lambda}^k)\mathbf{V}^{\mathsf{T}}$ were all used. Note that $\boldsymbol{\lambda}^k$ is a shorthand notation that means $[\boldsymbol{\lambda}^k]_i = \lambda_i^k$. The vector $\tilde{\mathbf{h}} \in \mathbb{R}^N$ is known as the frequency response of the filter and its i^{th} entry is given by

$$[\tilde{\mathbf{h}}]_i = \tilde{\mathsf{h}}(\lambda_i) = \sum_{k=0}^{K} h_k \lambda_i^k \tag{6}$$

where $\tilde{\mathsf{h}} : \mathbb{R} \to \mathbb{R}$ is a polynomial defined by the set of filter taps $\{h_k\}_{k=0}^{K}$. The function $\tilde{\mathsf{h}}(\cdot)$ depends only on the filter coefficients and not on the specific graph on which it is applied, and thus is valid for all graphs. The effect of filtering on a specific graph comes from instantiating $\tilde{\mathsf{h}}(\cdot)$ on the specific eigenvalues of that graph. The function $\tilde{\mathsf{h}}(\cdot)$ is denoted as the frequency response as well, and it will be clear from context whether we refer to the function $\tilde{\mathsf{h}}(\cdot)$ or to the vector $\tilde{\mathbf{h}}$ given in (6). Note that since $\tilde{\mathsf{h}}(\cdot)$ is an analytic function, it can be applied to the square matrix $\mathbf{S}$ so that $\tilde{\mathsf{h}}(\mathbf{S}) = \boldsymbol{H}^{\text{lsi}}(\mathbf{S})$.

In the case of the LSIGF, it is observed from (5) that the i^{th} entry of the frequency response of the output $[\tilde{\mathbf{y}}]_i = \tilde{y}_i$ is given by the elementwise multiplication

$$\tilde{y}_i = \tilde{\mathsf{h}}(\lambda_i)\tilde{x}_i. \tag{7}$$

This implies that the frequency response of the output $\tilde{y}_i$ is the elementwise multiplication of the frequency response of the filter $\tilde{\mathsf{h}}(\lambda_i)$ and the frequency response of the input $\tilde{x}_i$. This makes (7) the analogue of the convolution theorem for graph signals. Therefore, oftentimes the LSIGF in (2) is called graph convolution. It is observed from (7) that LSIGFs are capable of learning any type of frequency response (low-pass, high-pass, etc.), but that they are not able to create frequencies, i.e., if $\tilde{x}_i = 0$, then $\tilde{y}_i = 0$.

Unlike discrete-time signals, the frequency response of the LSIGF is not computed in the same manner as the frequency response of graph signals. More specifically, the frequency response of the filter $\tilde{\mathbf{h}}$ can be directly computed from the filter taps $\mathbf{h}$ by means of a Vandermonde matrix $\mathbf{\Lambda} \in \mathbb{R}^{N\times(K+1)}$ given by $[\mathbf{\Lambda}]_{ik} = \lambda_i^{k-1}$ as follows:

$$\tilde{\mathbf{h}} = \mathbf{\Lambda}\mathbf{h}. \tag{8}$$

This implies that the graph convolution is not commutative.

When considering NVGFs, as in (3), the convolution theorem (7) no longer holds. Instead, the frequency response of the output is given in the following proposition.

Proposition 1 (Frequency response of NVGF). *Let* $\mathbf{y} = \boldsymbol{H}^{\text{nv}}(\mathbf{S})\mathbf{x} = \sum_{k=0}^{K} \mathsf{diag}(\mathbf{h}^{(k)})\mathbf{S}^k\mathbf{x}$ *be the output of an arbitrary NVGF characterized by some filter tap matrix* $\mathbf{H} \in \mathbb{R}^{N\times(K+1)}$. *Then, the frequency response* $\tilde{\mathbf{y}} = \mathbf{V}^{\mathsf{T}}\mathbf{y}$ *of the output is given by*

$$\tilde{\mathbf{y}} = \mathbf{V}^{\mathsf{T}}\big(\mathbf{V} \circ (\mathbf{H}\mathbf{\Lambda}^{\mathsf{T}})\big)\tilde{\mathbf{x}} \tag{9}$$

where $\circ$ *denotes elementwise product of matrices.*

It is immediately observed that NVGFs are capable of generating new frequency content, even though they are linear.

Corollary 2. *If the matrix* $\mathbf{V}^{\mathsf{T}}\big(\mathbf{V} \circ (\mathbf{H}\mathbf{\Lambda}^{\mathsf{T}})\big)$ *is not diagonal, then the output exhibits frequency content that is not present in the input.*

To finalize the frequency analysis of NVGFs, we establish their Lipschitz continuity to changes in the underlying graph support, as decribed by the matrix $\mathbf{S}$. In what follows, we denote the spectral norm of a matrix $\mathbf{A}$ by $\|\mathbf{A}\|_2$.

Theorem 3 (Lipschitz continuity of the NVGF with respect to $\mathbf{S}$). *Let* $\mathcal{G}$ *and* $\hat{\mathcal{G}}$ *be two graphs with* N *nodes, described by GMDs* $\mathbf{S} \in \mathbb{R}^{N\times N}$ *and* $\hat{\mathbf{S}} \in \mathbb{R}^{N\times N}$, *respectively. Let* $\mathbf{H} \in \mathbb{R}^{N\times(K+1)}$ *be the coefficients of any NVGF. Given a constant* $\varepsilon > 0$, *if* $\|\hat{\mathbf{S}} - \mathbf{S}\|_2 \leq \varepsilon$, *it holds that*

$$\Big\|\big(\boldsymbol{H}^{\text{nv}}(\hat{\mathbf{S}}) - \boldsymbol{H}^{\text{nv}}(\mathbf{S})\big)\mathbf{x}\Big\|_2 \leq \varepsilon C\sqrt{N}(1+8N)\|\mathbf{x}\|_2 + \mathsf{O}(\varepsilon^2) \tag{10}$$

where $\boldsymbol{H}^{\text{nv}}(\mathbf{S})$ *and* $\boldsymbol{H}^{\text{nv}}(\hat{\mathbf{S}})$ *are the NVGF on* $\mathbf{S}$ *and* $\hat{\mathbf{S}}$, *respectively, and where* C *is the Lipschitz constant of the frequency responses at each node, i.e.,* $|\tilde{\mathsf{h}}_t(\lambda_j) - \tilde{\mathsf{h}}_t(\lambda_i)| \leq C|\lambda_j - \lambda_i|$ *for all* $i, j, t \in \{1, \ldots, N\}$.

Theorem 3 establishes the Lipschitz continuity of the NVGF filter with respect to the support matrix $\mathbf{S}$ (Lipschitz continuity with respect to the input $\mathbf{x}$ is immediately given for bounded filter taps) as long as the graphs $\mathbf{S}$ and $\hat{\mathbf{S}}$ are similar, i.e., $\varepsilon \ll 1$. The bound is proportional to this difference, ε, and to the shape of the frequency responses at each node through the Lipschitz constant C. It also depends on the number of nodes N, but it is fixed for given graphs with the same number of nodes. In short, Theorem 3 gives mild guarantees on the expected performance of the NVGF across a wide range of graphs $\hat{\mathbf{S}}$ that are close to the graph $\mathbf{S}$.

4. APPROXIMATING ACTIVATION FUNCTIONS

One of the main roles of activation functions in neural networks is to create low-frequency content that can be processed in a stable manner [15]. However, the way the nonlinearities create this frequency content is unknown and cannot be shaped. One alternative for tailoring the frequency creation process to the specific problem under study is to learn the NVGF filter taps (Sec. 5). However, doing so, implies that the number of learnable parameters depend on N which may lead to overfitting. In what follows we propose one method of designing, instead of learning, the NVGFs.

Problem statement. Assume that each data point $\mathbf{x}$ is a random graph signal with mean $\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}_x$, correlation matrix $\mathbf{R}_x = \mathbb{E}[\mathbf{x}\mathbf{x}^{\mathsf{T}}]$, and covariance matrix $\mathbf{C}_x = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu}_x)(\mathbf{x} - \boldsymbol{\mu}_x)^{\mathsf{T}}]$. The objective is to estimate a pointwise nonlinear function $\rho : \mathbb{R} \to \mathbb{R}$ such that $[\rho(\mathbf{x})]_i = \rho([\mathbf{x}]_i)$, using a NVGF-based estimator as $\hat{\mathbf{y}} = \boldsymbol{H}^{\text{nv}}(\mathbf{S})\mathbf{x} + \mathbf{c}$ for $\boldsymbol{H}^{\text{nv}}(\mathbf{S})$ as in (3) and $\mathbf{c} \in \mathbb{R}^N$. Given the random variable $\mathbf{y} = \rho(\mathbf{x})$, the aim is to find the filter taps $\mathbf{H} \in \mathbb{R}^{N\times(K+1)}$ that minimize the mean squared error (MSE)

$$\mathbf{H}^\star = \underset{\mathbf{H}\in\mathbb{R}^{N\times(K+1)}}{\arg\min}\ \mathbb{E}\big[\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2\big]. \tag{11}$$

Our particular focus is set on obtaining unbiased estimators.

Lemma 4 (Unbiased estimator). *Let* $\boldsymbol{\mu}_\rho = \mathbb{E}[\rho(\mathbf{x})]$. *The NVGF-based estimator is unbiased if and only if* $\mathbf{c} = \boldsymbol{\mu}_\rho - \boldsymbol{H}^{\text{nv}}(\mathbf{S})\boldsymbol{\mu}_x$.

From Lemma 4, the unbiased estimator is now

$$\hat{\mathbf{y}} = \boldsymbol{H}^{\text{nv}}(\mathbf{S})\big(\mathbf{x} - \boldsymbol{\mu}_x\big) + \boldsymbol{\mu}_\rho. \tag{12}$$

Therefore, the objective becomes finding the filter tap matrix $\mathbf{H} \in \mathbb{R}^{N\times(K+1)}$ for some fixed value of K that satisfies

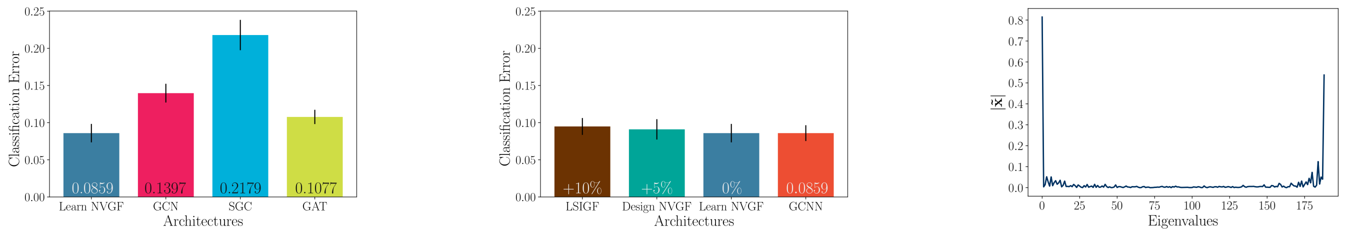
$$\mathbf{H}^\star = \underset{\mathbf{H}\in\mathbb{R}^{N\times(K+1)}}{\arg\min}\ \mathbb{E}\Big[\big\|\boldsymbol{H}^{\text{nv}}(\mathbf{S})(\mathbf{x}-\boldsymbol{\mu}_x) - \big(\rho(\mathbf{x}) - \boldsymbol{\mu}_\rho\big)\big\|_2^2\Big]. \tag{13}$$

Note that, due to the orthogonal nature of the GFT, minimizing (13) is equivalent to minimizing the difference of the corresponding frequency responses.

The optimal filter taps for each node, i.e., the rows $\mathbf{h}_i^\star \in \mathbb{R}^{K+1}$ of $\mathbf{H}^\star$ that solve (13), can be obtained by solving a linear system of equations as follows.

Proposition 5 (Optimal NVGF). *Let* $\mathbf{u}_i$ *denote the* i^{th} *row of* $\mathbf{V}$, $\mathbf{R}_i = \mathbf{\Lambda}^{\mathsf{T}}\mathsf{diag}(\mathbf{u}_i)\mathbf{V}^{\mathsf{T}}\mathbf{C}_x\mathbf{V}\mathsf{diag}(\mathbf{u}_i)\mathbf{\Lambda}$ *be the covariance matrix of the frequency response at node* v_i *for the input* $\mathbf{x}$, *and* $\mathbf{p}_i = \mathbf{\Lambda}^{\mathsf{T}}\mathsf{diag}(\mathbf{u}_i)\mathbf{V}^{\mathsf{T}}\mathbb{E}[(\rho(x_i) - \mu_{\rho i})(\mathbf{x} - \boldsymbol{\mu}_x)]$ *denote the correlation between the filtered signal and the target nonlinearity. Then, a set of filter taps* $\{\mathbf{h}_1^\star, \ldots, \mathbf{h}_N^\star\}$ *is optimal for* (13) *if and only if they solve the system of linear equations*

$$\mathbf{R}_i\mathbf{h}_i^\star = \mathbf{p}_i, \quad i \in \{1, \ldots, N\}. \tag{14}$$



(a) Performance Comparison (b) Role of frequency creation (c) Input frequencies

Fig. 1. Authorship attribution problem: Jane Austen. ((a)) Comparison between the Learn NVGF and three popular architectures (GCN [24], SGC [25] and GAT [26]). ((b)) Change with respect to a GCNN (15) when considering no activation function (LSIGF), a designed NVGF (16) and a learned NVGF (17). ((c)) A sample of the input frequencies to the architectures, where a large high-frequency component is observed.

From Proposition 5, it is immediate that only knowledge of the first and second moments of $\mathbf{x}$, and of the correlation between the input $\mathbf{x}$ and the output $\mathbf{y} = \rho(\mathbf{x})$, is required to solve for the optimal NVGF. These moments can be estimated from training data. Also observed is that the optimal filter taps for each node can be computed separately at each node.

5. GRAPH NEURAL NETWORK ARCHITECTURES USING NODE-VARIANT GRAPH FILTERS

A graph convolutional neural network (GCNN) Φ has a layered architecture [6, 23], where each layer applies a LSIGF as in (2), followed by a pointwise nonlinear activation function $\rho : \mathbb{R} \to \mathbb{R}$ applied to each node $[\rho(\mathbf{x})]_i = \rho([\mathbf{x}]_i)$ and

$$\mathbf{x}_0 = \mathbf{x},\ \mathbf{x}_\ell = \rho\big(\mathsf{H}_\ell^{\text{lsi}}(\mathbf{x}_{\ell-1}; \mathbf{S}, \mathbf{h}_\ell)\big),\ \mathbf{x}_L = \Phi(\mathbf{x}; \mathbf{S}, \mathcal{H}). \quad (15)$$

The LSIGF $\mathsf{H}_\ell^{\text{lsi}}(\cdot; \mathbf{S}, \mathbf{h}_\ell)$ is characterized by the specific filter taps $\mathbf{h}_\ell \in \mathbb{R}^{K_\ell+1}$. Note that the output of the GCNN is collected as the output of the last layer $\mathbf{x}_L = \Phi(\mathbf{x}; \mathbf{S}, \mathcal{H})$. This notation emphasizes that the input is $\mathbf{x}$, while the matrix $\mathbf{S}$ is given by the problem, and the filter taps $\mathcal{H} = \{\mathbf{h}_\ell\}_{\ell=1}^{L}$ are learned from data.

Nonlinear activation functions are used in GCNNs to enable them to learn nonlinear relationships between input and output. Additionally, theoretical results have found that they play a key role in creating frequency content that can be better processed by subsequent graph convolutional filters. As previously discussed, NVGFs are also capable of creating new frequency content, albeit in a linear manner. Therefore, by replacing the nonlinear activation functions by NVGFs, it is possible to decouple the contribution made by frequency creation from that made by the architecture's nonlinearity.

The first architecture proposed here is to use a designed NVGF in lieu of the activation function. That is, instead of using the nonlinear activation function ρ, an optimal NVGF filter designed as in Proposition 5 is used. This requires estimating the first and second moments of the NVGF input data. The architecture, herein termed "Design NVGF", is given by

$$\mathbf{x}_\ell = \mathsf{H}_\ell^{\text{nv}}\big(\mathsf{H}_\ell^{\text{lsi}}(\mathbf{x}_\ell; \mathbf{S}, \mathbf{h}_\ell); \mathbf{S}, \mathbf{H}_\ell^\star\big). \quad (16)$$

Note that, in this case, the filter taps $\mathbf{H}_\ell^\star \in \mathbb{R}^{N \times (K_\ell+1)}$ of the NVGF are obtained by Proposition 5, while the filter taps $\mathcal{H} = \{\mathbf{h}_\ell\}_{\ell=1}^{L}$ of the LSIGF are learned from data.

Alternatively, the filter taps of the NVGF replacing the nonlinear activation function can also be learned from data, together with the filter taps of the LSIGF:

$$\mathbf{x}_\ell = \mathsf{H}_\ell^{\text{nv}}\big(\mathsf{H}_\ell^{\text{lsi}}(\mathbf{x}_\ell; \mathbf{S}, \mathbf{h}_\ell); \mathbf{S}, \mathbf{H}_\ell\big) \quad (17)$$

where the filter taps to be learned are $\mathcal{H} = \{(\mathbf{h}_\ell, \mathbf{H}_\ell)\}_{\ell=1}^{L}$. This approach avoids the need to estimate first and second moments. Additionally, it allows the NVGF to learn how to create frequency content tailored to the application at hand, instead of just approximating the chosen nonlinear activation function. We note that while the increased number of parameters may lead to overfitting, this can be tackled by dropout. This architecture is termed "Learn NVGF".

6. NUMERICAL EXPERIMENTS

The objective of the numerical experiment is to isolate the impact that frequency creation has on the overall performance. To do this, we focus on the GSP problem of authorship attribution, which is a problem with signals known to have high-frequency content [5].

Problem statement. In the problem of authorship attribution, the goal is to determine whether a given text has been written by a certain author, relying on other texts that the author is known to have written. To this end, word adjacency networks (WANs) are leveraged. WANs are graphs that are built by considering function words (i.e., words without semantic meaning) as nodes, and considering their co-occurrences as edges [5]. As it happens, the way each author uses function words gives rise to a stylistic signature that can be used to attribute authorship. In what follows, we address this problem by leveraging previously constructed WANs as graphs, and the frequency count (histogram) of function words as the corresponding graph signals.

Dataset. For illustrative purposes, in what follows, works by Jane Austen are considered. Attribution of other 19th century authors can be found in the supplementary material. A WAN consisting of 189 nodes (function words) and 9812 edges is built from texts belonging to a given corpus considered to be the training set. These texts are partitioned into segments of approximately 1000 words each, and the frequency count of those 189 function words in each of the texts is obtained. These represent the graph signals $\mathbf{x}$ that are considered to be part of the training set. Each of these is assigned a label 1 to indicate that they have been authored by Jane Austen. An equal number of segments from other contemporary authors are randomly selected, and then their frequency count is computed and added to the training set with the label 0 to indicate that they have not been written by Jane Austen. The total number of labeled samples in the training set is 1464, of which 118 are left aside for validation. The test set is built analogously by considering other text segments that were not part of the training set (and thus, not used to build the WAN either), totaling 78 graph signals (half corresponding to texts authored by Jane Austen, and half corresponding to texts by other

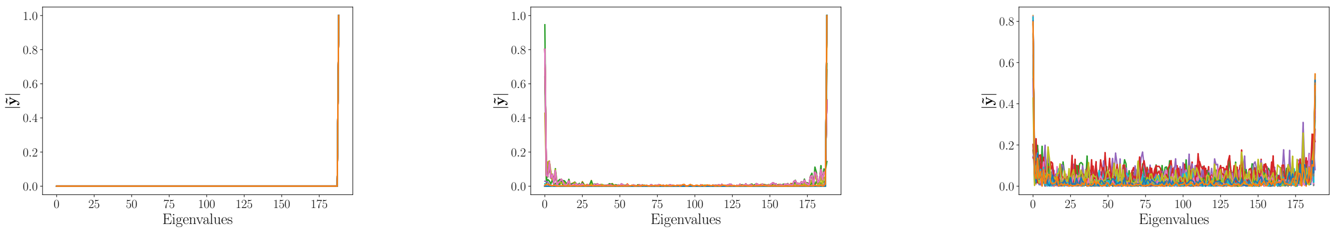

(a) LSIGF (b) GCNN (c) Learn NVGF

Fig. 2. Frequency responses to a single high-frequency input $\mathbf{x} = \mathbf{v}_N$. The frequency responses include the outputs of all $F = 64$ filters.

contemporary authors).

Architectures and training. For the first experiment, we compare the Learn NVGF architecture (17) with arguably three of the most popular non-GSP GNNs, namely, GCN [24], SGC [25], and GAT [26]. Note that the Learn NVGF is an entirely linear architecture, but one capable of creating frequencies due to the nature of the NVGF. The filter taps of both the LSIGF and the NVGF are learned from data. The other three architectures are nonlinear since they include a ReLU activation function after the first filtering layer. All architectures include a learnable linear readout layer. Dropout with a probability 0.5 is included after the first layer. All architectures are trained for 25 epochs with a batch size of 20, using the ADAM algorithm [30] with the forgetting factors 0.9 and 0.999, respectively, as well as a learning rate η. The value of the learning rate η, the number of hidden units F, and the number of filter taps K are chosen by exhaustive search over triplets (η, F, K) in the set $\{0.001, 0.005, 0.01\} \times \{16, 32, 64\} \times \{2, 3, 4\}$.

Experiment 1: Performance comparison. The objective of this first experiment is to illustrate that the performance of the Learn NVGF is comparable to the performance of popular (non-GSP) GNNs. The best results for each architecture are shown in Figure 1a, where the classification error was averaged over 10 random splits of texts that are assigned to the training and test sets. One third of the standard deviation is also shown. It is observed that the Learn NVGF architecture has a comparable performance. It is emphasized that the objective of this paper is not to achieve state-of-the-art performance, but to offer insight on the role of nonlinear activation functions in frequency creation and how this translates to performance, as discussed next.

Experiment 2: The role of frequency creation. For the second experiment, we compare the Learn NVGF of the previous experiment with (i) a simple LSIGF, (ii) a Design NVGF as in (16), and (iii) a GCNN as in (15). The same values of $(\eta = 0.001, F = 32, K = 3)$ are used for all architectures as a means of fixing all other variabiliity except for the nonlinearity/frequency creation. The results are shown in Figure 1b. Note that the GCNN in (15) can be interpreted as a stand-in for ChebNets [23], arguably the most popular GSP-based GNN architecture.

Discussion. First, note that the LSIGF, Design NVGF, and Learn NVGF architectures are linear, whereas the GCNN is not. The LSIGF, however, is not capable of creating frequencies, while the other three are, albeit through different mechanisms. Second, Figure 1b shows the percentage difference in performance with respect to the base architecture (the GCNN). Essentially, the difference in performance between the LSIGF and the Learn NVGF can be pinned down to the frequency creation, because both architectures are linear, while the difference between the Learn NVGF and the GCNN can be tied to the nonlinear nature of the GCNN. It is thus observed that the LSIGF performs considerably worse than the Learn NVGF and the GCNN, which perform the same. It is observed that the Design NVGF performs halfway between the GCNN and the LSIGF. The Design NVGF depends on the ability to accurately estimate the first and second moments from the data, and this has an impact on its performance. In any case, it is noted that this experiment suggests that the main driver of improved performance is the frequency creation and not necessarily the nonlinear nature of the GCNN.

From a qualitative standpoint, the average frequency response of the signals in the test set is shown in Figure 1c. Since the high-eigenvalue content is significant, it is expected that the ability to better process this content will impact the overall performance. This explains the relatively poor performance of the LSIGF. In Figure 2, we show the frequency response of the output for each of the three architectures (LSIGF, Learn NVGF, and GCNN) when the input has a single high frequency, i.e., $\mathbf{x} = \mathbf{v}_N$ so that $\tilde{\mathbf{x}} = \mathbf{e}_N$. Figure 2a shows that the output frequency response of the LSIGF exhibits a single frequency, the same as the input. Figure 2b shows that the output frequency response of the GCNN has content in all frequencies, but most notably a low-frequency peak appears. Figure 2c shows that the output frequency response of the Learn NVGF contains all frequencies, in a much more spread manner than the GCNN.

General observations. In the supplementary material, a similar analysis is carried out for 21 other authors. Additionally, it is noted that Jane Austen is representative of the largest group (consisting of 11 authors) where the Learn NVGF and the GCNN have similar performance and are better than the LSIGF. For 7 other authors, the Learn NVGF actually performs better than the GCNN. Finally, for the remaining 3 authors, there is no significant difference between the performance of the LSIGF, the GCNN, and the Learn NVGF, which implies that the high-frequency eigenvalue content is less significant for these authors.

7. CONCLUSION

The objective of this work was to study the role of frequency creation in GSP problems. To do so, nonlinear activation functions (which theoretical findings suggest give rise to frequency creation) are replaced by NVGFs, which are also capable of creating frequencies, but in a linear manner. In this way, frequency creation was decoupled from the nonlinear nature of activation functions. Numerical experiments show that the main driver of improved performance is frequency creation and not necessarily the nonlinear nature of GCNNs. As future work, we are interested in extending this frequency analysis to non-GSP related problems such as semi-supervised node classification or graph classification problems, which require a careful definition of a notion of frequency.

8. REFERENCES

[1] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, July 2017.

[2] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "Graphs, convolutions, and neural networks: From graph filters to graph neural networks," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 128–138, Nov. 2020.

[3] F. Monti, M. M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *31st Conf. Neural Inform. Process. Syst.* Long Beach, CA: Neural Inform. Process. Syst. Foundation, 4-9 Dec. 2017, pp. 3700–3710.

[4] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Trans. Signal Process.*, vol. 67, no. 1, pp. 97–109, 5 Nov. 2018.

[5] S. Segarra, M. Eisen, and A. Ribeiro, "Authorship attribution through function word adjacency networks," *IEEE Trans. Signal Process.*, vol. 63, no. 20, pp. 5464–5478, 30 June 2015.

[6] 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, 17 Dec. 2018.

[7] M. Eisen and A. Ribeiro, "Optimal wireless resource allocation with random edge graph neural networks," *IEEE Trans. Signal Process.*, vol. 68, pp. 2977–2991, 20 Apr. 2020.

[8] F. Gama and S. Sojoudi, "Distributed linear-quadratic control with graph neural networks," *Signal Process.*, 11 Feb. 2022, early access. [Online]. Available: http://doi.org/10.1016/j.sigpro.2022.108506

[9] D. Owerko, F. Gama, and A. Ribeiro, "Predicting power outages using graph neural networks," in *2018 IEEE Global Conf. Signal and Inform. Process.* Anaheim, CA: IEEE, 26-29 Nov. 2018, pp. 743–747.

[10] F. Gama, J. Bruna, and A. Ribeiro, "Stability properties of graph neural networks," *IEEE Trans. Signal Process.*, vol. 68, pp. 5680–5695, 25 Sep. 2020.

[11] H. Kenlay, D. Thanou, and X. Dong, "On the stability of graph convolutional neural networks under edge rewiring," in *46th IEEE Int. Conf. Acoust., Speech and Signal Process.* Toronto, ON: IEEE, 6-11 June 2021.

[12] R. Levie, W. Huang, L. Bucci, M. Bronstein, and G. Kutyniok, "Transferability of spectral graph convolutional neural networks," *arXiv:1907.12972v2 [cs.LG]*, 5 March 2020. [Online]. Available: http://arxiv.org/abs/1907.12972

[13] F. Gao, G. Wolf, and M. Hirn, "Geometric scattering for graph data analysis," in *36th Int. Conf. Mach. Learning*, vol. 97. Long Beach, CA: Proc. Mach. Learning Res., 9-15 June 2019, pp. 2122–2131.

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

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

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

[17] 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, 30 Apr. 2014.

[18] L. Ruiz, F. Gama, and A. Ribeiro, "Graph neural networks: Architectures, stability and transferability," *Proc. IEEE*, vol. 109, no. 5, pp. 660–682, May 2021.

[19] S. Segarra, A. G. Marques, and A. Ribeiro, "Optimal graph-filter design and applications to distributed linear networks operators," *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 4117–4131, 11 May 2017.

[20] S. Rey, V. Tenorio, S. Rozada, L. Martino, and A. G. Marques, "Deep encoder-decoder neural network architectures for graph output signals," in *53rd Asilomar Conf. Signals, Systems and Comput.* Pacific Grove, CA: IEEE, 3-6 Nov. 2019, pp. 225–229.

[21] S. Rey, V. Tenorio, S. Rozada, L. Martino, and A. G. Marques, "Overparametrized deep encoder-decoder schemes for inputs and outputs defined over graphs," in *28th Eur. Signal Process. Conf.* Amsterdam, the Netherlands: Eur. Assoc. Signal Process., 18-21 Jan. 2021, pp. 855–859.

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

[23] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *30th Conf. Neural Inform. Process. Syst.* Barcelona, Spain: Neural Inform. Process. Syst. Foundation, 5-10 Dec. 2016, pp. 3844–3858.

[24] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th Int. Conf. Learning Representations*, Toulon, France, 24-26 Apr. 2017, pp. 1–14.

[25] F. Wu, T. Zhang, A. H. de Souza Jr., C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *36th Int. Conf. Mach. Learning*, vol. 97. Long Beach, CA: Proc. Mach. Learning Res., 9-15 June 2019, pp. 6861–6871.

[26] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th Int. Conf. Learning Representations*, Vancouver, BC, 30 Apr.-3 May 2018, pp. 1–12.

[27] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, 11 Jan. 2013.

[28] 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.

[29] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.

[30] D. P. Kingma and J. L. Ba, "ADAM: A method for stochastic optimization," in *3rd Int. Conf. Learning Representations*, San Diego, CA, 7-9 May 2015, pp. 1–15.