Untrained Graph Neural Networks for Denoising

Samuel Rey, Student Member, IEEE, Santiago Segarra, Member, IEEE, Reinhard Heckel, Member, IEEE, and Antonio G. Marques, Senior Member, IEEE

Abstract—A fundamental problem in signal processing is to denoise a signal. While there are many well-performing methods for denoising signals defined on regular supports, such as images defined on two-dimensional grids of pixels, many important classes of signals are defined over irregular domains such as graphs. This paper introduces two untrained graph neural network architectures for graph signal denoising, provides theoretical guarantees for their denoising capabilities in a simple setup, and numerically validates the theoretical results in more general scenarios. The two architectures differ on how they incorporate the information encoded in the graph, with one relying on graph convolutions and the other employing graph upsampling operators based on hierarchical clustering. Each architecture implements a different prior over the targeted signals. To numerically illustrate the validity of the theoretical results and to compare the performance of the proposed architectures with other denoising alternatives, we present several experimental results with real and synthetic datasets.

Index Terms—Geometric Deep Learning, Graph Neural Networks, Graph Decoder, Graph Signal Denoising, Graph Signal Processing

I. INTRODUCTION

VAST amounts of data are generated and stored every day, propelling the deployment of data-driven solutions to address a wide variety of real-world problems. Unfortunately, the input data suffers from imperfections and is corrupted with noise, oftentimes associated with the data-collection process. Noisy signals appear in a gamut of applications, with examples including the processing of voice and images, the measurements in electric, social and transportation networks, or the monitoring of biological signals. As a result, signal denoising, which is the process of separating the signal from the noise, is a critical and ubiquitous task in contemporary data-science applications. While most existing works focus on the denoising of signals defined over regular domains (time and space), signals with irregular supports are becoming pervasive. Hence, designing (nonlinear) denoising schemes for signals defined over irregular domains arises as a worth-investigating problem. Examples of applications that benefit from reducing the amount of noise present in the data include processing signals defined over sensor networks, signal measured in the different regions of the brain, or signals related to protein segments. To name a few [1].

A versatile and tractable approach to handle information supported on irregular domains is to represent the structure of the domain as a graph, with nodes representing variables and edges encoding levels of similarity, influence, or statistical dependence among nodes. Successful examples of this approach can be found in the subareas of network analytics, machine learning over graphs, and graph signal processing (GSP) [2]–[4]. with graph neural networks (GNNs) and GSP being particularly relevant for the architectures presented in this paper [5], [6]. Since traditional data-processing architectures may incur difficulties learning the more complex structure present in many contemporary applications, GSP provides a principled approach to handle this issue [1], [4], [5]. Assuming that the structure of the signals can be modeled by a graph, GSP uses the information encoded in the graph topology to analyze, process, and learn from the data. As a result, it is not surprising that GSP has been successfully applied to design and analyze GNNs [6]–[9]. a class of neural network (NN) architectures that incorporate the graph topology information to enhance their performance when the data is composed of signals defined over a graph.

The importance of leveraging the graph influence when using deep non-linear architectures is reflected in the wide range of GNNs that co-exist in the literature, including graph convolutional NNs (GCNNs) [10]–[12], graph recurrent NNs [13], graph autoencoders [14]–[16], graph generative adversarial networks [17], [18], or simplicial NNs [19]–[21], to name a few. Incorporating the graph structure into deep non-linear models involves a wide range of options when designing the architecture. For example, GCNNs can be defined with or without pooling layers and the convolution over a graph can be implemented in several ways (vertex vs frequency), each leading to an architecture with different properties and performance. In fact, one of the key questions when designing a GNN is to decide the particular way in which the graph is incorporated into the architecture.

Motivated by the previous discussion, the goal of this work is twofold. First, we propose new graph-based NN architectures to denoise (smooth) graph signals, with the difference between the architectures residing in how they incorporate the information encoded in the graph. Second, we provide theoretical guarantees for the denoising capabilities of this approach, and show that it is directly influenced by the properties of the underlying graph. The mathematical analysis, performed on particular instances of these architectures, provides guarantees on their denoising performance under specific assumptions for the original signal and its underlying graph. In addition, we numerically validate the denoising performance of our method for more general scenarios than those covered by our theory, illustrating that the proposed graph-aware untrained
architectures can effectively denoise graph signals.

Since the presented architectures are untrained NNs, only one noisy observation is needed to recover the original signal and no training data is used. The underlying assumption is that, due to their architecture, the NNs are capable of learning the structure of the original signal faster than the noise. Hence, the denoising process for each observed signal is carried out by fitting the weights for a few iterations. This same phenomenon has been observed to hold true in non-graph deep learning architectures. In the context of denoising, the optimization of the overparametrized architecture is stopped early, so that overfitting to the noise is avoided.

To incorporate the topology of the graph, the first architecture multiplies the input at each layer by a fixed (non-learnable) graph filter [22], which can be seen as a generalization of a (low-pass) message passing operation. The second architecture performs graph upsampling operations to progressively increase the size of the input until it matches the size of the observed signal. The upsampling operators are based on hierarchical clustering algorithms [15], [23]–[25] so that, in contrast with [26], matrix inversions are not required, avoiding the related numerical issues. Our work is substantially different from [15], [16], which deal with graph encoder-decoder architectures. On top of our theoretical analysis and extensive numerical simulations, additional differences to prior work are that: (a) our graph decoder is an untrained network, and thus, it does not need a training phase; (b) we only require a decoder-like architecture for denoising graph signals, so it is not necessary to jointly design and train two different architectures as done in [15], [16].

Contributions and outline. In summary, the contributions of the paper are the following: (i) we present two new overparametrized and untrained GNNs for solving graph-signal denoising problems; (ii) mathematical analysis is conducted for each architecture offering bounds for their performance, improving our understanding about non-linear architectures and the influence of incorporating graph structure into NNs; and (iii) the proposed architectures are evaluated and compared to other denoising alternatives through numerical experiments carried out with synthetic and real-world data.

The remainder of the paper is organized as follows. Section II reviews related works dealing with graph-signal denoising. Section III explains fundamental concepts leveraged along the paper. Section IV formally introduces the problem at hand and presents our general approach. Sections V and VI detail the proposed architectures and provide the mathematical analysis for each of them. Numerical experiments are presented in Section VII and concluding remarks are provided in Section VIII.

A. Related works

Untrained NNs are a family of architectures that, by carefully incorporating prior information of the signals in the architecture, enable the recovery of signals without the need of training over large (or any) datasets [27]–[30]. In [27], it is shown that fitting a standard convolutional autoencoder to only one noisy signal using early stopping enables the effective denoising of an image. For this approach to work, it is critical that the signal class (images) matches the NN architecture (two-dimensional convolutional NN with particular filters).

Previous approaches to the graph-signal denoising task included a graph-regularization term that promoted desired properties on the estimated signals [31]. Some existing works minimize the graph total variation pushing the signal value at neighboring nodes to be close [31], [32]. A related approach assumes that the signals are smooth on the graph and add a regularization parameter based on the quadratic form of the graph Laplacian [33]. Also, in [34], the authors propose a spectral graph trilateral filter as a regularizer, based on the prior assumption that the gradient is smooth over the graph. Although these alternatives rely on imposing some notion of smoothness on the original graph signal, the actual relation between the signal and the graph may be of a different nature. Furthermore, the actual prior may be more complex than that represented by linear and quadratic terms.

More recently, non-linear solutions for denoising graph signals have been proposed to tackle the aforementioned issues. In [35], a median graph filter [36], [37] is used to denoise a set of time-varying graph signals defined over dynamic graphs. The idea is to use a smooth non-linear (median) operator that combines values of neighboring nodes, leveraging both spatial and temporal adjacency relations. A different non-linear approach is followed in [26], where a graph autoencoder is trained to recover the denoised signals. To change the size of the graph, the autoencoder relies on Kron reduction operations [38]. However, since the Kron reduction is based on the inverse of a submatrix of the graph Laplacian, it could fall into numerical issues if the submatrix is singular. Moreover, both architectures need several observations to recover the noiseless signals. The median graph filter approach is constrained to the case where a time series of graph signals is given, while the latter needs a high enough number of observations to train its network parameters before being able to denoise the observed signals.

II. PROCESSING ARCHITECTURES FOR GRAPH SIGNALS

In this section, we introduce notation, present the fundamentals of GSP, and discuss GNNs. A key theme throughout this section is to formalize how the properties of a given signal depend on the supporting graph, which is critical for our denoising methods.

A. Fundamentals of GSP

Let \( G = (\mathcal{V}, \mathcal{E}) \) denote an undirected graph, where \( \mathcal{V} \) is the set of nodes with cardinality \( N \), and \( \mathcal{E} \) is the set of links such that \((i, j)\) and \((j, i)\) belong to \( \mathcal{E} \) if nodes \( i \) and \( j \) are connected. The set \( \mathcal{V}_i := \{j | (i,j) \in \mathcal{E} \} \) denotes the neighborhood of node \( i \). For a given graph \( \mathcal{G} \), the (sparse) symmetric adjacency matrix \( A \in \mathbb{R}^{N \times N} \) has non-zero entries \( A_{ij} \) only if \((i,j) \in \mathcal{E}\). If \( \mathcal{G} \) is unweighted, its entries \( A_{ij} \) are binary. If the graph is weighted, then the value of \( A_{ij} \) captures the strength of the

\[ A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{E} \text{ and } A_{ij} \text{ is binary} \\ \text{weight of edge } (i,j) & \text{otherwise} \end{cases} \]

Although our theoretical results assume that the graph is undirected, the architectures and algorithms proposed in this paper can tackle signals defined on directed graphs [39].
link between nodes $i$ and $j$. In this paper, we focus on the processing of graph signals which are defined on $V$. Graph signals can be represented as a vector $x = [x_1, \ldots, x_N]^T \in \mathbb{R}^N$, where the $i$-th entry represents the value of the signal at node $i$. Since the signal $x$ is defined on $G$, the core assumption of GSP is that the properties of $x$ depend on the topology of $G$. For instance, consider a graph that encodes similarity. If the value of $A_{ij}$ is high, then one expects the signal values $x_i$ and $x_j$ to be similar or closely related.

**Graph-shift operator (GSO).** The GSO is defined as an $N \times N$ matrix $S$ whose entry $S_{ij}$ can be non-zero only if $i = j$ or $(i, j) \in E$. Common choices for $S$ are the adjacency matrix $A$, or its degree normalized alternative $\tilde{A} := D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, where $D = \text{diag}(A1)$ is the degree matrix, $1$ is the vector of all ones, and $\text{diag}(\cdot)$ is the diagonal operator that turns a vector into a diagonal matrix. Another common choice is the combinatorial graph Laplacian $\mathbf{L}$, defined as $L := D - A$ [4]. The GSO accounts for the topology of the graph and, at the same time, it represents a linear transformation that can be computed locally. Specifically, if $y = [y_1, \ldots, y_N]^T$ is defined as $y = Sx$, then node $i$ can compute $y_i$ provided that it has access to the values of $x_j$ at its neighbors $j \in V_i$. We also assume that the GSO is diagonalizable so that there exists an orthonormal matrix $V$ and a diagonal matrix $\Lambda$, both of size $N \times N$, such that $S = \mathbf{V} \Lambda \mathbf{V}^T$.

**Graph filtering.** Graph filters, an important tool of GSP, are linear operators $\mathbb{R}^N \rightarrow \mathbb{R}^N$ that can be expressed as a polynomial of the GSO of the form
\[
\mathbf{H} := \sum_{m=0}^{M-1} h_m S^m, \tag{1}
\]
where $\mathbf{H}$ is the graph filter, $h_m$ are the graph filter coefficients, and $M \leq N$ [22]. Since $S^m$ encodes the $m$-hop neighborhoods of the graph $G$, graph filters can be used to diffuse input graph signals $x$ across the graph as $y = \sum_{m=0}^{M-1} h_m S^m x = \mathbf{H} x$. Because graph filters are capable of diffusing signals across $(M-1)$-hop neighborhoods, they are widely used to generalize the convolution operation to signals defined over graphs. Furthermore, since the graph filter $\mathbf{H}$ is a polynomial on $S$, it follows that both matrices have the same eigenvectors $V$.

**Frequency representation.** The frequency domain of graph signals and filters is determined by the eigendecomposition of the GSO. More precisely, the frequency representation of the graph signal $x$ is given by the $N$-dimensional vector $\hat{x} = V^T x$, with $V$ acting as the graph Fourier transform (GFT) [40]. Similarly, the frequency response of graph filter $\mathbf{H}$ can be defined as $\hat{H} = \text{diag}(V^T \mathbf{H} V)$, that is, an $N$-dimensional vector collecting the eigenvalues of $\mathbf{H}$ [22], [40].

A graph signal (filter) is said to be bandlimited (low-pass) if its frequency domain representation $\hat{x}$ satisfies that $\hat{x}_k = 0$ for $k > K$, where $K \leq N$ is referred to as the bandwidth of the signal $x$. If $x$ is bandlimited with bandwidth $K$ it holds that
\[
x = V_K \hat{x}_K, \tag{2}
\]
with $\hat{x}_K = [\hat{x}_1, \ldots, \hat{x}_K]$ collecting the active frequency components and $V_K$ collecting the corresponding $K$ eigenvectors. In other words, the bandlimited representation states that the original $N$-dimensional signal $x$ lies in a subspace of reduced dimensionality related to the spectrum of the graph. This reduced-dimensionality representation, which can be generalized to graph filters as well, has been shown to bear practical relevance in real-world datasets and can be exploited in denoising and other inverse problems [41].

**B. Fundamentals of GNNs**

Generically, we represent a GNN using a parametric non-linear function $f_\Theta(Z G) : \mathbb{R}^{N_0} \times F^{(0)} \rightarrow \mathbb{R}^N$ that depends on the graph $G$. The parameters of the architecture are collected in $\Theta$, and the matrix $Z \in \mathbb{R}^{N_0} \times F^{(0)}$ represents the input of the network. Although there are many possibilities for defining a specific GNN, a broad range of such architectures can be represented by recursively applying a graph-aware linear transformation followed by an entry-wise non-linearity. Then, a generic deep architecture $f_\Theta(Z G)$ with $L$ layers can be described as
\[
\hat{Y}^{(l)} = \mathcal{T}_\Theta^{(l)} \left\{ Y^{(l-1)}(G) \right\}, \quad 1 \leq l \leq L, \tag{3}
\]
\[
Y_{ij}^{(l)} = g^{(l)} \left( Y_{ij}^{(l-1)} \right), \quad 1 \leq l \leq L, \tag{4}
\]
where $Y^{(0)} = Z$ and $y = Y^{(L)}$ denote the input and output of the architecture, $\mathcal{T}_\Theta^{(l)}(G) : \mathbb{R}^{N^{(l-1)} \times F^{(l-1)}} \rightarrow \mathbb{R}^{N^{(l)} \times F^{(l)}}$ is a graph-aware linear transformation performed at layer $l$, $\Theta^{(l)} \in \mathbb{R}^{F^{(l-1)} \times F^{(l)}}$ are the parameters that define such a transformation, $g^{(l)} : \mathbb{R} \rightarrow \mathbb{R}$ is a scalar nonlinear transformation (e.g., a ReLU function), which is oftentimes omitted in the last layer. Moreover, $N^{(l)}$ and $F^{(l)}$ represent the number of nodes and features at layer $l$, $\Theta = \{\Theta^{(l)}\}_{l=1}^L$ collects all the parameters of the architecture, and $y$ denotes the output of the GNN. Note that although the function $f_\Theta(Z G)$ has been introduced as generating output signals defined in $\mathbb{R}^N$, which is the case of interest for this paper, it can be easily adapted to output graph signals with more than one feature.

**III. GNNs for Graph-Signal Denoising**

We now formally introduce the problem of graph-signal denoising within the GSP framework, and present our approach to tackle it using untrained GNN architectures. Given the graph $G$, let us consider the observed graph signal $x \in \mathbb{R}^N$, which is a noisy version of the original graph signal $x_0 \in \mathbb{R}^N$. With $n \in \mathbb{R}^N$ being a noise vector, the relation between $x$ and $x_0$ is
\[
x = x_0 + n. \tag{5}
\]
Then, the goal of graph-signal denoising is to remove as much noise as possible from the observed signal $x$ to estimate the original signal $x_0$, which is performed by exploiting the information encoded in $G$.

A traditional approach for the graph-signal denoising task is to solve an optimization problem of the form
\[
x_0 = \arg\min_{x_0} \|x - x_0\|^2 + \alpha R(x_0 | G). \tag{6}
\]
The first term promotes fidelity to the signal observations, the regularizer $R(\cdot | G)$ promotes denoised signals with desirable
Algorithm 1: Proposed graph-signal denoising method

**Inputs**: \( x \) and \( G \)

**Outputs**: \( x_0 \) and \( \Theta(x) \)

1. Set \( f_{\Theta}(Z|G) \) as explained in Section IV or V
2. Generate \( Z \) from iid zero-mean Gaussian distribution
3. Initialize \( \Theta(0) \) from iid zero-mean Gaussian
4. for \( t = 1 \) to \( T \) do
5. \( \Theta(x) = \Theta(T) \)
6. \( \hat{x}_0 = f_{\Theta(x)}(Z|G) \)
7. end

Regarding the specific implementation of the untrained network \( f_{\Theta}(Z|G) \), there are multiple possibilities for selecting the linear and non-linear transformations \( T_{\Theta(t)} \) and \( g^{(t)} \) defined in equations (3) and (4), respectively. Since we are dealing with the denoising of analog signals, we set the entrywise non-linearity \( g^{(t)} \) to be the ReLU operation, defined as \( \text{ReLU}(x) = \max(0, x) \), and focus on the design of the linear transformation, which is responsible for incorporating the structure of the graph. The two following sections postulate the implementation of two particular linear transformations \( T^{(t)}_{\Theta(t)} \) (each giving rise to a different GNN) and analyze the resulting architectures.

IV. GRAPH CONVOLUTIONAL GENERATOR

Our first architecture to address the graph-signal denoising task is a graph-convolutional generator (GCG) network that incorporates the topology of the graph into the NN pipeline via vertex-based graph convolutions. To formally define the GCG architecture, we select the normalized adjacency matrix \( \tilde{A} = D^{-1/2}AD^{-1/2} \) as the GSO \( S \). Then, leveraging the fact that convolutions of a graph signal on the vertex domain can be represented by a graph filter \( H \in \mathbb{R}^{N \times N} \) [22], we define the linear transformation for the convolutional generator as (cf. (3))

\[
T_{\Theta(t)} \{ Y^{(t-1)} | G \} = HY^{(t-1)} \Theta^{(t)}. \tag{9}
\]

Remember that the \( F^{(t-1)} \times F^{(t)} \) matrix \( \Theta^{(t)} \) collects the learnable weights of the \( \ell \)-th layer, and the graph filter \( H \) is given by (1) with its coefficients \( \{ h_m \}_{m=0}^{M-1} \) fixed a priori so that \( H \) is a low-pass graph filter of degree \( M \). Using the linear transformation defined in (9), the output of the GCG with \( L \) layers is given by the recursion

\[
Y^{(t)} = \text{ReLU}(HY^{(t-1)} \Theta^{(t)}), \quad \text{for} \quad \ell = 1, \ldots, L-1, \tag{10}
\]

\[
y^{(L)} = HY^{(L-1)} \Theta^{(L)}, \tag{11}
\]

where \( Y^{(0)} = Z \) and the ReLU is not applied in the the last layer of the architecture.

With the proposed linear transformation, the GCG learns to combine the features within each node by fitting the weights of the matrices \( \Theta^{(t)} \) while the graph filter \( H \) interpolates the signal by mixing features from \( M-1 \) neighborhoods. Therefore, since \( H \) is a low-pass graph filter, the GCG promotes smooth outputs and, thus, a smooth denoised estimate \( \hat{x}_0 \). In addition, for a given layer, despite the linear mapping being from \( N \times F^{(t-1)} \rightarrow N \times F^{(t)} \), we limit the degrees of freedom by imposing a Kronecker structure so only \( N^2 + F^{(t)}F^{(t-1)} \) parameters are involved (cf. (9)), and only \( F^{(t)}F^{(t-1)} \) of all the parameters need to be learned since \( H \) is given.

Although we define the GCG using a graph convolutional layer, there is an important difference when comparing it with other GCNNs. In some GCNNs, the parameters of the graph filter are learned, but in the proposed architecture the graph filter is fixed so it promotes desired properties on the estimate \( \hat{x}_0 \). Moreover, from the polynomial definition of \( H \) it can be noted that the fixed graph filter may be interpreted as a generalization of the message passing procedure [42], a typical approach for performing graph convolutions in NNs.

In the remainder of the section, we adopt some simplifying assumptions to provide theoretical guarantees on the denoising capability of the GCG, and then we rely on numerical evaluation to demonstrate that the results also hold in more general settings.
A. Guaranteed denoising with the GCG

To formally prove that the proposed architecture can successfully denoise the observed graph signal $x$, we consider a two-layer GCG given by

$$f_\Theta(Z|\mathcal{G}) = \text{ReLU}(HZ\Theta^{(1)}\theta^{(2)}),$$

where $\Theta^{(1)} \in \mathbb{R}^{F \times F}$ and $\theta^{(2)} \in \mathbb{R}^F$ are the learnable coefficients. With $F$ denoting the number of features, we consider the overparametrized regime where $F \geq 2N$, and hence, minimizing over $\theta^{(2)} = 0$.

We start by noting that scaling the $i$-th entry of $\theta^{(2)}$ is equivalent to scaling the $i$-th column of $\Theta^{(1)}$, so that, without loss of generality, we can set the weights to $\theta^{(2)} = b$, where $b$ is a vector of size $F$ with half of its entries set to $1/\sqrt{F}$ and the other half to $-1/\sqrt{F}$. Furthermore, since $Z$ is a random matrix of dimension $N \times F$, the column space of $Z$ spans $\mathbb{R}^N$, and hence, minimizing over $Z\Theta^{(1)}$ is equivalent to minimizing over $\Theta \in \mathbb{R}^{N \times F}$. With these considerations in place, the optimization over (7) can be equivalently performed replacing the two-layer GCG described in (12) by its simplified form

$$f_\Theta(H) = f_\Theta(Z|\mathcal{G}) = \text{ReLU}(H\Theta)b.$$

Note that we replaced $f_\Theta(Z|\mathcal{G})$ with $f_\Theta(H)$ since the graph influence is modeled by the graph filter $H$, and the influence of the matrix $Z$ is absorbed by the learnable weights $\Theta$.

The denoising capability of the two-layer architecture is related to the eigendecomposition of its expected squared Jacobian $\mathcal{X}$ (29). However, to understand which signals can be effectively denoised with the proposed architecture, we need to connect the spectral domain of the expected squared Jacobian with the spectrum of the graph, given by the eigenvectors of the GSO.

To that end, we next compute the expected squared Jacobian of the two-layer architecture in (15). Denote as $\mathcal{J}_\Theta(H) \in \mathbb{R}^{N \times NF}$ the Jacobian matrix of $f_\Theta(H)$ with respect to $\Theta$, which is given by

$$\mathcal{J}_\Theta(H) = \begin{bmatrix} b_1 H^T \text{diag} \left( \text{ReLU}'(H\theta_1) \right) \\ \vdots \\ b_F H^T \text{diag} \left( \text{ReLU}'(H\theta_F) \right) \end{bmatrix} \in \mathbb{R}^{NF \times N},$$

where $\theta_i$ represents the $i$-th column of $\Theta$, and $\text{ReLU}'$ is the derivative of the ReLU, which is the step function. Then, define the $N \times N$ expected squared Jacobian matrix as

$$\mathcal{X} := \mathbb{E}_\Theta[\mathcal{J}_\Theta(H) \mathcal{J}_\Theta(H)^T].$$

Taking the expectation of (14) with respect to the parameters $\Theta$, and leveraging the results from (43 Section 3.2), we obtain that the matrix $\mathcal{X}$ is given by

$$\mathcal{X} = 0.5 \left( 11^T - \frac{1}{\pi} \text{arccos} \left( C^{-1} H^2 C^{-1} \right) \right) \otimes HH^T,$$

where $\otimes$ represents the Hadamard (entry-wise) product, $\text{arccos}(\cdot)$ is computed entry-wise, $h_i$ represents the $i$-th column (row) of $H$, $C = \text{diag}(||h_1||^2, \ldots, ||h_N||^2)$ is a normalization term so that $C^{-1} H^2 C^{-1}$ is the autocorrelation of the graph filter $H$.

Since $\mathcal{X}$ is symmetric and positive (semi) definite, it has an eigendecomposition $\mathcal{X} = W\Sigma W^T$. Here, the columns of the orthonormal matrix $W = [w_1, \ldots, w_N]$ are the $N$ eigenvectors, and the nonnegative eigenvalues in the diagonal matrix $\Sigma$ are assumed to be ordered as $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_N$.

After defining the two-layer GCG $f_\Theta(H)$ and its expected square Jacobian $\mathcal{X}$, we formally analyze its performance when denoising bandlimited graph signals. This is particularly relevant given the importance of (approximate) bandlimited graph signals both from analytical and practical points of view (4). For the sake of clarity, we first introduce the main result (Theorem 1) and then we detail a key intermediate result (Lemma 1) that provides additional insight.

Formally, consider the $K$-bandlimited graph signal $x_0$ as described in (2), and let the architecture $f_\Theta(H)$ have a sufficiently large number of features $F$:

$$F \geq \left( \frac{\sigma_1^2}{\sigma_2^2} \right)^{26} \xi^{-8N},$$

where $\xi \in (0, 2(\log(N/\phi))^{-1/2})$ is an error tolerance parameter for some prespecified $\phi$. Then, for a specific set of graphs that is introduced later in the section (cf. Assumption 1), if we solve (47) running gradient descent with a step size $\eta \leq \frac{1}{\sigma_1^2}$, the following result holds (see Appendix A).

**Theorem 1.** Let $f_\Theta(H)$ be the network defined in equation (13), and assume it is sufficiently wide, i.e., it satisfies condition (17) for some error tolerance parameter $\xi$. Let $x_0$ be a $K$-bandlimited graph signal spanned by the eigenvectors $V_K$, and let $w_i$ and $\sigma_i$ be the $i$-th eigenvector and eigenvalue of $\mathcal{X}$. Let $n$ be the noise present in $x$, and set $\phi$ and $\epsilon$ to small positive numbers. Then, for large enough $N$ ($N > N_\epsilon$, $\delta$), the error for each iteration $t$ of gradient descent with stepsize $\eta$ used to fit the architecture is bounded as

$$||x_0 - f_{\Theta}(H)||_2 \leq (1 - \eta \sigma_K^2)^t \left[ \left(1 - \eta \sigma_K^2 \right) + \left(1 - \eta \sigma_2^2 \right) \right] \frac{||x_0||_2}{\xi} + \frac{\xi ||x||_2}{\epsilon} + \sum_{i=1}^{N} \left[ (1 - \eta \sigma_i^2)^t - 1 \right] (w_i^T n)^2,$$

with probability at least $1 - e^{-F^2 - \phi - \epsilon}$.

As explained next, the fitting (denoising) bound provided by the theorem first decreases and then increases with the number of iterations $t$. To be more precise, let us analyze separately each of the three terms in the right hand side of (18). The first term captures the part of the signal $x_0$ that is fitted after $t$ iterations while accounting for the misalignment of the eigenvectors $V_K$ and $W_K$. This term decreases with $t$ and, since $\delta$ can be made arbitrary small for sufficiently large enough graphs (cf. Lemma 1), vanishes for moderately low values of $t$. The second term is an error term that is negligible if the network is sufficiently wide so that $\xi$ can be chosen to be sufficiently while condition (17) remains satisfied. Finally, the third term, which depends on the noise present in each of the spectral components of the squared Jacobian $(w_i^T n)^2$, grows with $t$. More specifically, if the $\sigma_i$ associated with a spectral component is very small, the term $(1 - \eta \sigma_i^2)$ is close to 1 and, hence, the noise power in the $i$-th frequency
will be small. Only when \( t \) grows very large the coefficient \((1 - \eta \sigma_i^2)^t\) vanishes and the \( i \)-th frequency component of the noise is fitted. As a result, if the filter \( \mathbf{H} \) is designed such that eigenvalues of the squared Jacobian satisfy that \( \sigma_K \gg \sigma_{K+1} \), then there will be a range of moderate-to-high values of \( t \) for which: i) the first term is zero and ii) only the \( K \) strongest components of the noise have been fitted, so that the third term can be approximated as \( \sqrt{\sum_{i=1}^{K} (w_i^T n)^2} \). Clearly, as \( t \) grows larger, the coefficient \((1 - \eta \sigma_i^2)^t - 1\) will also be close to one for \( i > K \), meaning that additional components of the noise will be fitted as well, deteriorating the performance of the denoising architecture. This implies that if the optimization algorithm is stopped before \( t \) grows too large, the original signal is fitted along with the noise that aligns with the signal, but not the noise present in other components.

In other words, Theorem 1 not only characterizes the performance of the two-layer GNN, but also illustrates that, if early stopping is adopted, our overparametrized architecture is able to effectively denoise the bandlimited graph signal.

Note that a critical step to attain Theorem 1 is to relate the eigenvectors of \( \mathbf{X} \) with those of the GSO \( \mathbf{S} \), denoted as \( \mathbf{V} \). To achieve this, we assume that \( \mathbf{S} = \mathbf{A} + \alpha \mathbf{1} \mathbf{1}^T \) is random and provide high-probability bounds between the leading eigenvectors of \( \mathbf{A} \) and \( \mathbf{X} \). More specifically, consider a graph \( \mathcal{G} \) drawn from a stochastic block model (SBM) \([44]\) with \( K \) communities. Also, denote by \( \mathcal{M}(\mathbf{A}) \) the SBM with expected adjacency matrix \( \mathbf{A} = \mathbb{E}[\mathbf{A}] \), and by \( \beta_{\min} \) the minimum expected degree \( \beta_{\min} := \min_i [\mathbf{A}]_i \). Given some \( \rho > 0 \), we define as \( \mathcal{M}_N(\rho) \) the class of SBMs \( \mathcal{M}(\mathbf{A}) \) with \( N \) nodes for which \( \beta_{\min} = \omega(\ln(N/\rho)) \), where \( \omega(\cdot) \) denotes the (conventional) asymptotic dominance. In this context, we consider the following assumption.

**Assumption 1.** The model \( \mathcal{M}(\mathbf{A}) \) from which \( \mathbf{A} \) is drawn satisfies \( \mathcal{M}(\mathbf{A}) \in \mathcal{M}_N(\rho) \).

Intuitively, it is assumed that the expected minimum degree of the SBM increases as the number of nodes grows. Under these conditions, the following result holds.

**Lemma 1.** Let the matrix \( \mathbf{X} \) be defined as in \([10]\), set \( \epsilon \) and \( \delta \) to small positive numbers, and denote by \( \mathbf{V}_K \) and \( \mathbf{W}_K \) the \( K \) leading eigenvectors in the respective eigendecompositions of \( \mathbf{A} \) and \( \mathbf{X} \). Under Assumption 1, there exists an orthonormal matrix \( \mathbf{Q} \) and an integer \( N_{\epsilon,\delta} \) such that, for \( N > N_{\epsilon,\delta} \), the bound

\[
\|\mathbf{V}_K - \mathbf{W}_K \mathbf{Q}\|_F \leq \delta,
\]

holds with probability at least \( 1 - \epsilon \).

The proof is provided in Appendix B. Lemma 1 guarantees that, if the size of the graph is big enough, the difference between the subspaces spanned by the leading eigenvectors of \( \mathbf{A} \) and \( \mathbf{X} \) is bounded, becoming arbitrarily small as the number of nodes increases. An inspection of \([10]\) reveals that the result in Lemma 1 is not entirely unexpected. Indeed, since \( \mathbf{H} \) is a polynomial in \( \mathbf{S} \), so is \( \mathbf{H}^2 \). This implies that \( \mathbf{V} \) are also the eigenvectors of \( \mathbf{H}^2 \), and because \( \mathbf{H}^2 \) appears twice on the right hand side of \([10]\), a relationship between the eigenvectors of \( \mathbf{X} \) and \( \mathbf{V} \) can be anticipated. However, the presence of the Hadamard product and the (non Lipschitz continuous) non-linearity \( \arccos \) renders the exact analysis of the eigenvectors a challenging task. Consequently, we resorted to a stochastic framework in deriving Lemma 1.

**B. Analyzing the deep GCG**

While for convenience, the previous section focused on analyzing the GCG architecture with \( L = 2 \) layers, in practice we often work with a larger number of layers. In this section, we provide numerical evidence showing that the relation between matrices \( \mathbf{A} \) and \( \mathbf{X} \) described in Lemma 1 also holds when \( L > 2 \).

To that end, Figure 1 shows the pairs of eigenvectors \( \mathbf{v}_i \) and \( \mathbf{w}_i \) for the indexes \( i = \{1, 3, 10, 64\} \), for a given graph \( \mathcal{G} \) drawn from an SBM with \( N = 64 \) nodes and 4 communities. The GCG is composed of \( L = 5 \) layers and, to obtain the eigenvectors of the squared Jacobian matrix, the Jacobian is computed using the autograd functionality of PyTorch. The nodes of the graph are sorted by communities, i.e., the first \( N_1 \) nodes belong to the first community and so on. It can be clearly seen that, even for moderately small graphs, the leading eigenvectors of \( \mathbf{A} \) and \( \mathbf{X} \) are almost identical, becoming more dissimilar as the eigenvectors are associated with smaller eigenvalues. It can also be observed how leading eigenvectors have similar values for entries associated with nodes within the same community. Moreover, Figure 2 depicts the matrix product \( \mathbf{V}^T \mathbf{W} \), where it is observed that the \( K = 4 \) leading eigenvectors of both matrices are orthonormal. The presented numerical results strengthen the argument that the analytical results obtained for the two-layer case can be extrapolated to deeper architectures.

In addition to using an architecture with only two layers, another important assumption of Lemma 1 is that the graph \( \mathcal{G} \) is drawn from an SBM. This assumption facilitates the derivation of a bound relating the spectra of \( \mathbf{A} \) and \( \mathbf{X} \) (i.e., the subspaces spanned by the eigenvectors \( \mathbf{V}_K \) and \( \mathbf{W}_K \)). The numerical experiments reported in Figure 3 illustrate that such a relation also holds for other type of graphs. The figure has 12 panels (3 rows and 4 columns). Each of the rows corresponds to a different graph, namely: 1) a realization of a small-world (SW) graph \([45]\) with \( N = 150 \) nodes, 2) the Zachary’s Karate graph \([46]\) with \( N = 34 \) nodes, and 3) a graph of \( N = 316 \) weather stations across the United States \([47]\). Each of the three first columns correspond to an \( N \times N \) matrix, namely: 1) the normalized adjacency matrix \( \mathbf{A} \), 2) \( \mathbf{H}_2 \), the squared version of a low pass graph filter with \( \mathbf{S} = \mathbf{A} \) and whose coefficients are drawn from a uniform distribution and set to unit \( \ell_1 \) norm, and 3) the squared Jacobian matrix \( \mathbf{X} \). Although we may observe some similarity between \( \mathbf{A} \) and \( \mathbf{X} \), the relation between \( \mathbf{X} \) and the graph \( \mathcal{G} \) becomes apparent when comparing the matrices \( \mathbf{H}_2 \) and \( \mathbf{X} \). The matrix \( \mathbf{H}_2 \) is a random graph filter used in the linear transformation of the convolutional generator \( f_{\Theta}(\mathbf{H}) \), and it is clear that the vertex connectivity pattern of \( \mathbf{X} \) is related to that of \( \mathbf{H}_2 \). Since \( \mathbf{X} \) and \( \mathbf{H}_2 \) are closely related and we know that the eigenvectors of \( \mathbf{H}_2 \) and those of \( \mathbf{A} \) are the same, we expect \( \mathbf{W} \) (the eigenvectors of \( \mathbf{X} \)) and \( \mathbf{V} \) (the eigenvectors of \( \mathbf{A} \)) to
be related as well. To verify this, the fourth column of Figure 3
represents $V_K^T W_K$, i.e., the pairwise inner products of the $K$
leading eigenvectors of $\tilde{A}$ and those of $\mathcal{X}$. It can be observed
that the $K$ leading eigenvectors are close to orthogonal, which
means that the relation observed in the vertex domain carries
over to the spectral domain and $V_K$ and $W_K$ expand the same
subspace. As a result, our GCG will be capable of denoising
a signal $x$ that lives in the subspace spanned by $V_K$ for all
the considered graphs.

To summarize, the presented results illustrate that the
analytical characterization provided in Section IV-A, which
considered a two-layer GCG operating over SBM graphs,
carries over to more general setups.

V. GRAPH UPSAMPLING DECODER

The GCG architecture presented in Section IV incorporated
the topology of $G$ via the vertex-based convolutions imple-
mented by the graph filter $H$ with $S = \tilde{A}$, in this section,
we introduce the graph decoder (GD) architecture, a new
graph-aware denoising NN that incorporates the topology of $G$
via a (nested) collection of graph upsampling operators [48].
Specifically, we propose the linear transformation for the GD
denoiser to be given by

$$
J_{\Theta^{(\ell)}} \{ Y^{(\ell-1)} | g \} = U^{(\ell)} Y^{(\ell-1)} \Theta^{(\ell)},
$$

(19)

where $U^{(\ell)} \in \mathbb{R}^{N^{(\ell)} \times N^{(\ell-1)}}$, with $N^{(\ell)} \geq N^{(\ell-1)}$, are
graph upsampling matrices to be defined soon. Note that, compared
to (9), the graph filter $H$ is replaced with the upsampling
operator $U^{(\ell)}$ that depends on $\ell$. Adopting the proposed linear
transformation, the output of the GD with $L$ layers is given
by the recursion

$$
Y^{(\ell)} = \text{ReLU}(U^{(\ell)} Y^{(\ell-1)} \Theta^{(\ell)}), \quad \text{for } \ell = 1, ..., L-1,
$$

(20)

$$
Y^{(L)} = U^{(L)} Y^{(L-1)} \Theta^{(L)},
$$

(21)

where the ReLU is also removed from the last layer.

Similarly to the GCG, the proposed GD learns to combine
the features within each node, with the interpolation of the
signals being controlled by the graph upsampling operators
$\{U^{(\ell)}\}_{\ell=1}^L$. The size of the input $N^{(0)}$ is now a design
parameter that will determine the implicit degrees of freedom
of the architecture. Note that, from the GSP perspective,
the input feature matrix $Y^{(\ell-1)} \in \mathbb{R}^{N^{(\ell-1)} \times F^{(\ell-1)}}$
represents $F^{(\ell-1)}$ graph signals, each of them defined over a graph
$G^{(\ell-1)}$ with $N^{(\ell-1)}$ nodes. Therefore, even though the input
$Y^{(0)} = Z$ is still a random white matrix across rows and
columns, since $N^{(\ell)} \geq N^{(\ell-1)}$, the dimensionality of the input
is progressively increasing.

When compared to the GCG, the smaller dimensionality
of the input $Z$ endows the GD architecture with less degrees
of freedom, rendering the architecture more robust to noise.
Furthermore, instead of relying on graph filters, the graph in-
formation is included through the graph upsampling operators
$U^{(\ell)}$. Clearly, the method used to design the graph upsampling
matrices, which is the subject of the next section, will have
an impact on the type of graph signals that can be efficiently
denoised using the GD architecture.

A. Graph upsampling operator from hierarchical clustering

Regular upsampling operators have been successfully used
in NN architectures to denoise signals defined on regular domains [29].
While the design of upsampling operators in regular grids is straightforward,
when the signals at hand are defined on irregular domains the problem becomes sub-
stantially more challenging. The approach that we put forth
in this paper is to use agglomerative hierarchical clustering
methods [23]–[25] to design a graph upsampling operator that
leverages the graph topology. These methods take a graph as
an input and return a dendrogram; see Figure 3. A dendrogram
can be interpreted as a rooted-tree structure that shows differ-
ent clusters at the different levels of resolution $\nu$. At the finest
resolution ($\nu = 0$) each node forms a cluster of its own. Then,
as $\nu$ increases, nodes start to group together (agglomerate)
in bigger clusters and, when the resolution becomes large (coarse)
ought, all nodes end up being grouped in the same cluster.
For each graph (rows), it can be seen that there is a relation between matrices $P$ of the graph filter $P = 0$ algorithm and the resulting graphs with 2, 4, 7 and 14 nodes.

Fig. 3: Illustrating the matrices $\tilde{A}$, $H^2$, $X$, and $V_K^T W_K$, shown in columns 1, 2, 3, and 4, respectively, for different types of graphs. The rows 1, 2, and 3 present a small world graph, the Zachary’s Karate graph, and the weather stations graph. The graph filter $H^2$ is created as a square graph filter with coefficients drawn from a uniform distribution and set to unit $\nu$. We leverage these parent-children relations to define the two clusters.

Fig. 4: Dendrogram of an agglomerative hierarchical clustering algorithm and the resulting graphs with 2, 4, 7 and 14 nodes.

By cutting the dendrogram at $L + 1$ resolutions, including $\nu = 0$, we obtain a collection of node sets with parent-child relationships inherited by the refinement of clusters. Since we are interested in performing graph upsampling, note that the dendrogram is interpreted from left to right. This can be observed in the example shown in Figure 4 where the three red nodes in the second graph ($\nu = 10$, layer $\ell = 1$) are children of the red parent in the coarsest graph ($\nu = 12$, layer $\ell = 0$). We leverage these parent-children relations to define the membership matrices $P^{(\ell)} \in \{0, 1\}^{N^{(\ell)} \times N^{(\ell-1)}}$, where the entry $P_{ij}^{(\ell)} = 1$ only if the $i$-th node in layer $\ell$ is the child of the $j$-th node in layer $\ell - 1$. Moreover, the clusters at layer $\ell$ can be understood as nodes of a graph $G^{(\ell)}$ with $N^{(\ell)}$ nodes and adjacency matrix $A^{(\ell)}$, which represents a coarser-resolution version of the original graph $G$. There are several ways of defining $A^{(\ell)}$ based on the original adjacency matrix $A$. While our architecture does not focus on a particular form, in the simulations we set $A_{ij}^{(\ell)} \neq 0$ only if, in the original graph $G$, there is at least one edge between nodes belonging to the cluster $i$ and nodes from cluster $j$. In addition, the weight of the edge depends on the number of existing edges between the two clusters.

With the definition of the membership matrix $P^{(\ell)}$, and letting $\tilde{A}^{(\ell)}$ denote the degree-normalized version of the adjacency matrix $A^{(\ell)}$, the upsampling operator of the $\ell$-th layer is given by

$$U^{(\ell)} = \left(\gamma I + (1 - \gamma) \tilde{A}^{(\ell)}\right)P^{(\ell)},$$

where $\gamma \in [0, 1]$ is a pre-specified constant. Notice that $U^{(\ell)}$ in (22) copies the signal value from the parents to the children by applying matrix $P^{(\ell)}$ and, then, every children performs a convex combination between this value and the average signal value of its neighbors. Therefore, the design of $U^{(\ell)}$ conveys a notion (prior) of smoothness on the targeted graph signals, since we are promoting that nodes descending from the same parent have similar (related) values.

Because the membership matrices $P^{(\ell)}$ are designed using a clustering algorithm over $G$, and the matrices $\tilde{A}^{(\ell)}$ capture how
strongly connected the clusters of layer \( \ell \) are in the original graph, these two matrices are responsible for incorporating the information of \( \mathcal{G} \) into the upsampling operators \( \mathbf{U}^{(\ell)} \). Furthermore, we remark that the upsampling operator \( \mathbf{U}^{(\ell)} \) can be reinterpreted as the application of \( \mathbf{P}^{(\ell)} \) followed by the application of a graph filter

\[
\hat{\mathbf{H}}^{(\ell)} = \gamma \mathbf{I} + (1 - \gamma) \hat{\mathbf{A}}^{(\ell)},
\]

which uses \( \hat{\mathbf{A}}^{(\ell)} \) as the GSO, and sets the filter coefficients as \( h_0 = \gamma \) and \( h_1 = 1 - \gamma \).

\[\text{B. Guaranteed denoising with the GD}\]

As we did for the GCG, our goal is to theoretically characterize the denoising performance of the GNN architecture defined by (20) and (22). To achieve this goal, we replicate the approach implemented in Section IV-A. We first derive the matrix \( \mathbf{X} \) and provide theoretical guarantees when denoising a \( K \)-bandlimited graph signal with the GD. Then, to gain additional insight, we detail the relation between the subspace spanned by the eigenvectors \( \mathbf{W} \) and the spectral domain of the GSO. This relation is key in deriving the theoretical analysis.

We start by introducing the two-layer GD

\[f_{\Theta}(\mathbf{Z}|\mathcal{G}) = \text{ReLU}(\mathbf{UZ}\Theta^{(1)})\theta^{(2)}.\]  

(24)

Upon following a reasoning similar to that provided after (13), optimizing the previous architecture is equivalent to optimizing its simplified version

\[f_{\Theta}(\mathbf{U}) = f_{\Theta}(\mathbf{Z}|\mathcal{G}) = \text{ReLU}(\mathbf{U}\Theta)b.\]

(25)

An important difference with respect to the GCG presented in (13) is that the matrix \( \Theta \) has a dimension of \( N^{(0)} \times F \), so it spans \( \mathbb{R}^{N^{(0)}} \) instead of \( \mathbb{R}^N \). Since \( N^{(0)} \leq N \), the smaller subspace spanned by the weights of the GD renders the architecture more robust to fitting noise, but, on the other hand, the number of degrees of freedom to learn the graph signal of interest are reduced. As a result, the alignment between the targeted graph signals and the low-pass vertex-clustering architecture becomes more important.

The expected squared Jacobian \( \mathbf{X} = \mathbb{E}_{\Theta}[\mathbf{J}_{\Theta}(\mathbf{U})][\mathbf{J}_{\Theta}(\mathbf{U})^T] \) is obtained following the procedure used to derive (16), arriving at the expression

\[
\mathbf{X} = 0.5 \left( \mathbf{11}^T - \frac{1}{\pi} \arccos(\hat{\mathbf{C}}^{-1}\mathbf{U}\mathbf{U}^T\hat{\mathbf{C}}^{-1}) \right) \odot \mathbf{UU}^T,
\]

(26)

where \( \mathbf{u}_i \) represents the \( i \)-th row of \( \mathbf{U} \), and \( \hat{\mathbf{C}} = \text{diag}([\|\mathbf{u}_1\|_2, \ldots, \|\mathbf{u}_N\|_2]) \) is a normalization matrix.

Then, let \( \mathbf{x}_0 \) be a \( K \)-bandlimited graph signal and let \( f_{\Theta}(\mathbf{U}) \) have a number of features \( F \) satisfying (17). If we solve (17) running gradient descent with a step size \( \eta \leq \frac{1}{\sigma^2} \), the following result holds.

\[\text{Theorem 2. Let } f_{\Theta}(\mathbf{U}) \text{ be the network defined in equation (25). Consider the conditions described in Theorem 7 and let } N^{(0)} \text{ match the number of communities } K \text{ by stacking more layers we perform the upsampling of the input signal in a progressive manner and, at the same time, we add more non-linearities, which helps alleviating the rank constraint related to the input size } N^{(0)}. \text{ In the absence of non-linear functions, the maximum rank of the weights would be}\]

The proof of the theorem is analogous to the one provided in Appendix A but exploiting Lemma 2 instead of Lemma 1. Lemma 2 is fundamental in attaining Theorem 2 and is presented later in the section.

Theorem 2 formally establishes the denoising capability of the GD when \( \mathbf{x}_0 \) is a \( K \)-bandlimited graph signal and \( K = N^{(0)} \) matches the number of communities in the SBM graph. When compared with the GCG, the smaller dimensionality of the input \( \mathbf{Z} \), and thus the smaller rank of the matrix \( \Theta \), constrains the learning capacity of the architecture, making it more robust to the presence of noise. However, this additional robustness also implies that the architecture is more sensitive to model mismatch, since its capacity to learn arbitrary signals is smaller. Intuitively, the GD represents an architecture tailored for a more specific family of graph signals than the GCG. Moreover, employing the GD instead of the GCG has a significant impact on the relation between the subspaces spanned by \( \mathbf{V}_K \) and \( \mathbf{W}_K \).

To establish the new relation between \( \mathbf{V}_K \) and \( \mathbf{W}_K \), assume that the adjacency matrix is drawn from an SBM \( \mathcal{M}(\mathcal{A}) \) with \( K \) communities such that \( \mathcal{M}(\mathcal{A}) \in \mathcal{M}_N(\rho) \), so that the SBM follows Assumption 1. In addition, set the size of the latent space to the number of communities so \( N^{(0)} = K \). Under this setting, the counterpart to Lemma 1 for the case where \( f_{\Theta}(\mathbf{U}) \) is a GD architecture follows.

\[\text{Lemma 2. Let the matrix } \mathbf{X} \text{ be defined as in (26), set } \epsilon \text{ and } \delta \text{ to small positive numbers, and denote by } \mathbf{V}_K \text{ and } \mathbf{W}_K \text{ the } K \text{ leading eigenvectors in the respective eigendecompositions of } \hat{\mathbf{A}} \text{ and } \mathbf{X}. \text{ Under Assumption 1 there exist an orthonormal matrix } \mathbf{Q} \text{ and an integer } N_{\epsilon,\delta} \text{ such that for } N > N_{\epsilon,\delta} \text{ the bound}\]

\[\|\mathbf{V}_K - \mathbf{W}_K \mathbf{Q}\|_F \leq \delta,\]

holds with probability at least \( 1 - \epsilon \).

\[\text{Lemma 2 asserts that the difference between the subspaces spanned by } \mathbf{V}_K \text{ and } \mathbf{W}_K \text{ becomes arbitrarily small as the size of the graph increases. The proof is provided in Appendix C and the intuition behind it arises from the fact that the upsampling operator can be understood as } \mathbf{U} = \mathbf{H} \mathbf{P}, \text{ where } \mathbf{H} \text{ is a graph filter of the specific form described in (23). Remember that } \mathbf{P} \text{ is a binary matrix encoding the cluster in the layer } \ell - 1 \text{ to which the nodes in the layer } \ell \text{ belong. Since we are only considering two layers, and we have that } N^{(0)} = K, \text{ the matrix } \mathbf{P} \text{ is encoding the node-community membership of the SBM graph and, hence, the product } \mathbf{P} \mathbf{P}^T \text{ is a block matrix with constant entries matching the block pattern of } \hat{\mathbf{A}}. \text{ As shown in the proof, this property can be leveraged to bound the eigendecomposition of } \hat{\mathbf{A}} \text{ and } \mathbf{X}.\]

\[\text{C. Analyzing the deep GD}\]

The deep GD composed of \( L > 2 \) layers can be constructed following the recursion presented in (20) and (21). In this case, by stacking more layers we perform the upsampling of the input signal in a progressive manner and, at the same time, we add more non-linearities, which helps alleviating the rank constraint related to the input size \( N^{(0)} \). In the absence of non-linear functions, the maximum rank of the weights would be
Deep Decoder experiments and the architectures is available on GitHub.

To that end, we generate an SBM graph with $N$ nodes and $K = 4$ communities, and define 3 different signals: (i) “Signal”: a piece-wise constant signal $x_0$ with the value of each node being the label of its community; (ii) “Noise”: zero-mean white Gaussian noise with unit variance; and (iii) “Signal + Noise”: a noisy observation $x = x_0 + n$ where the noise present a normalized power of 0.1. Figures 5a and 5b show the normalized mean squared error (MSE) obtained for each realization as

$$
\|x - x_0\|^2_2/\|x_0\|^2_2.
$$

The mean is computed for 100 realizations of graphs and graph signals. The normalized power of the noise present in the data is the median of all the realizations. The random graph models the eigenvector error goes to 0 as $N$ increases, with the only exception of the connected caveman graph. This helps to illustrate that, although the conditions assumed for Lemma 1 and Lemma 2 focus on the specific setting of the SBM, the results can be applied to a wider class of graphs, motivating thus the extension of the proposed theorems to more general settings as a future line of work.

VI. NUMERICAL RESULTS

This section presents different experiments to numerically validate the theoretical claims introduced in the paper, and to illustrate the denoising performance of the GCG and the GD. The experiments are carried out using synthetic and real-world data, and the proposed architectures are compared to other graph-signal denoising alternatives. The code for the experiments and the architectures is available on GitHub. For hyper-parameter settings and implementation details the interested reader is referred to the online available code.

A. Denoising capability of graph untrained architectures

The goal of the experiment shown in Figures 5a and 5b is to illustrate that the proposed graph untrained architectures are capable of learning the structured original signal $x_0$ faster than the noise, which is one of the core claims of the paper. To that end, we generate an SBM graph with $N = 64$ nodes and $K = 4$ communities, and define 3 different signals: (i) “Signal”: a piece-wise constant signal $x_0$ with the value of each node being the label of its community; (ii) “Noise”: zero-mean white Gaussian noise $n$ with unit variance; and (iii) “Signal + Noise”: a noisy observation $x = x_0 + n$ where the noise present a normalized power of 0.1. Figures 5a and 5b show the normalized mean squared error (MSE) obtained for each realization as

$$
\|x - x_0\|^2_2/\|x_0\|^2_2.
$$

The mean is computed for 100 realizations of the noise as the number of epochs increases. The reported error is the normalized MSE of the estimated signal $x_0$, and the figure shows the median of values of 100 realizations of graphs and graph signals.

The normalized power of the noise present in the data is 0.1. Graphs are drawn from an SBM with $N = 64$ nodes and 4 communities, and the graph signals are generated as: (i) a zero-mean white Gaussian noise with unit variance (“Rand”); (ii) a bandlimited signal using the $K$ leading eigenvectors of $\mathbf{X}$ as base (“J”); (iii) a bandlimited graph signal using the $K$ leading eigenvectors of $\mathbf{A}$ as base (“BL”); and (iv) a diffused white (“DW”) signal created as $y = \text{med}(\mathbf{H}w|\mathcal{G})$, where $w$ is a white vector whose entries are sampled from $\mathcal{N}(0, 1)$, $\mathbf{H}$ is a low-pass graph filter, and $\text{med}(\cdot|\mathcal{G})$ represents the graph-aware median operator such that the value of the node $i$ is the median of its neighborhood $[35]-[37]$. The results in Figure 5c show that the best denoising error is obtained when the signal is composed of just a small number of eigenvectors, and the performance deteriorates as the bandwidth (i.e., the number of leading eigenvectors that span the subspace where the signal lives) increases, obtaining the worst result when the signal is.

B. Denoising synthetic data

We now proceed to comment on the denoising performance of the proposed architectures with synthetic data. The usage of synthetic signals allows us to study how the properties of the noiseless signal influence the quality of the denoised estimate.

The first experiment, shown in Figure 6a, studies the error of the denoised estimate obtained with the 2-layer GCG as the number of epochs increases. The reported error is the normalized MSE of the estimated signal $x_0$, and the figure shows the median values of 100 realizations of graphs and graph signals.

The normalized power of the noise present in the data is 0.1. Graphs are drawn from an SBM with $N = 64$ nodes and 4 communities, and the graph signals are generated as: (i) a zero-mean white Gaussian noise with unit variance (“Rand”); (ii) a bandlimited signal using the $K$ leading eigenvectors of $\mathbf{X}$ as base (“J”); (iii) a bandlimited graph signal using the $K$ leading eigenvectors of $\mathbf{A}$ as base (“BL”); and (iv) a diffused white (“DW”) signal created as $y = \text{med}(\mathbf{H}w|\mathcal{G})$, where $w$ is a white vector whose entries are sampled from $\mathcal{N}(0, 1)$, $\mathbf{H}$ is a low-pass graph filter, and $\text{med}(\cdot|\mathcal{G})$ represents the graph-aware median operator such that the value of the node $i$ is the median of its neighborhood $[35]-[37]$. The results in Figure 6a show that the best denoising error is obtained when the signal is composed of just a small number of eigenvectors, and the performance deteriorates as the bandwidth (i.e., the number of leading eigenvectors that span the subspace where the signal lives) increases, obtaining the worst result when the signal is.

https://github.com/reysam93/Graph_Deep_Decoder
generated at random. This result is aligned with the theoretical claims since it is assumed that the signal $x_0$ is bandlimited. It is also worth noting that the architecture also achieves a good denoising error with the “DW” model, showcasing that the GCG is also capable of denoising other types of smooth graph signals.

Next, Figure 6b compares the performance of the 2-layer GCG (“2L-GCG”), the deep GCG (“GCG”) and the deep GD (“GD”) with the baseline models introduced in Section III which are the total variation (“TV”), Laplacian regularization (“LR”) and bandlimited model (“BL”). In this setting, the graphs are SBM with $N = 256$ nodes and 8 communities, and the signals are bandlimited with a bandwidth of 8. Since the “BL” model with $K = 8$ captures the actual generative model of the signal $x_0$, it achieves the best denoising performance. However, it is worth noting that the GCG obtains a similar result, outperforming the other alternatives. Moreover, the benefits of using the deep GCG instead of the 2-layer architecture are apparent, since it achieves a better performance in fewer epochs.

On the other hand, Figure 6c illustrates a similar experiment but with the graph signals generated as “DW”. Under this setting, it is clear that the GD outperforms the other alternatives, showcasing that it is more robust to the presence of noise when the signals are aligned with the prior implicitly captured by the GD architecture.

Fig. 6: Median MSE when denoising a graph signal as a function of the number of epochs. a) The 2-layer GCG is used to denoise different families of signals. b) Performance comparison between total variation, Laplacian regularization, bandlimited models, the 2-layer GCG, the deep GCG, and the deep GD, when the signals are bandlimited. c) Counterpart of b) for the case where signals are diffused white.
frequency components (“BL”), a graph-aware median operator such that the value of \( x_i \) is the median of its neighborhood (“MED”) \(^{[35]}\), and a GCNN. It can be observed that the GD is more robust to the presence of noise, since it outperforms the other alternatives and achieves a mean MSE of 0.049 when the noise power attains a value of 0.5. Moreover, note that the GCG outperforms the GCNN showcasing the advantage of using a fixed graph filter instead of learning the filter parameters. In the absence of noise, the GCG outperforms the other alternatives, including the GD. This illustrates that the GCG can be interpreted as a less regularized architecture than the GD.

**VII. Conclusion**

In this paper, we faced the relevant task of graph-signal denoising. To approach this problem, we presented two over-parametrized and untrained GNNs and provided theoretical guarantees on the denoising performance of both architectures when denoising \( K \)-bandlimited graph signals under some simplifying assumptions. Moreover, we numerically illustrated that the proposed architectures are also capable of denoising graph signals in more general settings. The key difference between the two architectures resided in the linear transformation that incorporates the information encoded in the graph. The GCG employs fixed (non-learnable) low-pass graph filters to model convolutions in the vertex domain, promoting smooth estimates. On the other hand, the GD relies on a nested collection of graph upsampling operators to progressively increase the input size, limiting the degrees of freedom of the architecture, and providing more robustness to noise. In addition to the aforementioned analysis, we tested the validity of the proposed theorems and evaluated the performance of both architectures with real and synthetic datasets, showcasing a better performance than other classical and non-linear methods for graph-signal denoising.

**APPENDIX A**

**Proof of Theorem 1**

Let \( x_0 \) be a \( K \) bandlimited graph signal as described in \(^{[2]}\), which is spanned by the \( K \) leading eigenvectors of the graph \( V_K \), with \( x_0 \) denoting its frequency representation. Denote as \( \bar{x}_0 = W_K Q x_0 \) the bandlimited signal using \( W_K \) as basis and whose frequency response is also \( x_0 \). Let \( Q \) be an orthonormal matrix that aligns the subspaces spanned by \( V_K \) and \( W_K \), and note that \( x_0 \) can be interpreted as recovering \( x_0 \) from its frequency response using \( W_K \) instead of \( V_K \). Also note that \( x_0 - \bar{x}_0 = (V_K - W_K Q) x_0 \) represents the error between the signal \( x_0 \) and its approximation inside the subspace spanned by \( W_K \). With these definitions in place, we have from \(^{[29]}\) Theorem 3] with probability at least \( 1 - e^{-\frac{F^2}{2}} - \delta \) that

\[
\| x_0 - f_{\Theta(t)}(Z|G) \|_2 \leq \| \Psi x_0 \|_2 + \xi \| x_0 \|_2
\]

with \( \Psi := W(I_N - \eta \Sigma^2)^T W^T \), and \( I_N \) the \( N \times N \) identity matrix. However, note that the error bound for the term \( \| \Psi x_0 \|_2 \) provided in \(^{[29]}\) does not apply since \( x_0 \) is not spanned by \( W_K \). Accordingly, we further bound this term as

\[
\| \Psi x_0 \|_2 \leq \| \Psi (x_0 + x_0 - \bar{x}_0) \|_2
\]

\[
= \| \Psi_K \bar{x}_0 + \Psi (V_K - W_K Q) x_0 \|_2
\]

\[
\leq \| \Psi_K \bar{x}_0 \|_2 + \| \Psi (V_K - W_K Q) x_0 \|_2
\]

\[
\leq \| \Psi_K \|_2 \| \bar{x}_0 \|_2 + \| \Psi \|_2 \| V_K - W_K Q \|_F \| x_0 \|_2
\]

\[
= \left( \| \Psi_K \|_2 + \| \Psi \|_2 \right) \| x_0 \|_2
\]

\[
\leq \left( \| \Psi_K \|_2 + \| \Psi \|_2 \right) \| x_0 \|_2
\]

\[
\leq \left( \| \Psi_K \|_2 + \| \Psi \|_2 \right) \| x_0 \|_2
\]

\[
\leq \left( \| \Psi_K \|_2 + \| \Psi \|_2 \right) \| x_0 \|_2
\]

Here, \( \Psi_K := W_K (I_K - \eta \Sigma^2)^T W_K^T \), and \( \Sigma_K \) represents a diagonal matrix containing the first \( K \) leading eigenvalues \( \sigma_k \). We have that \( (i) \) follows from \( x_0 \) being bandlimited in \( W_K \), so \( \Psi x_0 = \Psi_K \bar{x}_0 \). Then, \( (ii) \) follows from the triangle inequality, and \( (iii) \) from the \( \ell_2 \) norm being submultiplicative and using the Frobenius norm as an upper bound for the \( \ell_2 \) norm. In \( (iv) \) we apply the result of Lemma 1 which holds with probability at least \( 1 - \epsilon \) because \( N > N_{\epsilon, \delta} \), and the fact that, since both \( W_K \) and \( V_K \) are orthonormal matrices, we have that \( \| x_0 \|_2 = \| \bar{x}_0 \|_2 = \| x_0 \|_2 \). We obtain \( (v) \) from the largest eigenvalues present in \( \Psi_K \) and \( \Psi \).

Finally, replacing \(^{[28]} \) in \(^{[27]} \) the proof concludes.

**APPENDIX B**

**Proof of Lemma 1**

Define \( \tilde{A} := A := \mathbb{E}[A] = \mathbb{E}[D]^{-\frac{1}{2}} \mathcal{A} \mathbb{E}[D]^{-\frac{1}{2}} \) and let \( \chi \) be given by \(^{[16]} \). Denote by \( \mathcal{H} \) a graph filter defined as a polynomial of the expected adjacency matrix \( \tilde{A} \), and let \( \chi \) be the expected squared Jacobian using the graph filter \( \mathcal{H} \), i.e.,

\[
\chi = 0.5 \left( \frac{11 - \pi \arccos(C^{-1} \mathcal{H}^2 C^{-1})} {\pi} \right) \mathcal{H}^2,
\]

where \( C \) is the counterpart of \( C \) in \(^{[16]} \), but using \( \mathcal{H} \) instead of \( H \). Given the following eigendecompositions \( \mathcal{A} = V \Lambda V^T \), \( \chi = W \Sigma W^T \), \( \tilde{A} = V \Lambda V^T \), and \( \chi = W \Sigma W^T \), for arbitrary orthonormal matrices \( T \) and \( R \), we have that

**Bounding \( \| V_K - W_K Q \|_F \)**

From the definition of an SBM, it follows that \( \mathcal{A} = \mathbb{E}[A] = B \Omega B^T \), where \( B \in \{0, 1\}^{N \times K} \) is an indicator matrix encoding the community to which each node belongs, and \( \Omega \) is a \( K \times K \) matrix encoding the link probability between the communities of the graph. Therefore, \( \tilde{A} \) and \( \chi \) are both block matrices whose blocks coincide with the communities in the SBM. This implies that the eigenvectors associated with non-zero eigenvalues must span the columns of \( B \). Hence, the leading eigenvectors must be related by an orthonormal transformation, from where it follows that, given \( T \), we can always find \( R \) such that

\[
\| V_K T - W_K R \|_F = 0.
\]
Bounding $\|V_K - \tilde{V}_K T\|_F$. Under Assumption [1] as it is shown in [51], with probability at least $1 - \rho$ we have that
\[ \|\tilde{A} - \bar{A}\| \leq 3\sqrt{3\ln(4N/\rho)/\beta_{\min}}. \]
(32)

Denoting as $\lambda_i$ the $i$-th eigenvalue collected in $\bar{A}$, i.e. $\lambda_i = \bar{A}_{ii}$, we combine the concentration in (32) with the Davis-Kahan theory [52] to obtain that there exists an orthonormal matrix $T$ such that
\[ \|V_K - \tilde{V}_K T\|_F \leq \sqrt{8K}/\lambda_K - \lambda_{K+1} \|\tilde{A} - \bar{A}\|_F \leq 3\sqrt{8K}/\lambda_K \sqrt{3\ln(4N/\rho)/\beta_{\min}}, \]
(33)

where we note that, since $\bar{A}$ follows an SBM, then $\lambda_i = 0$ for all $i > K$.

Since $\beta_{\min} = \omega(\ln(N/\rho))$, we obtain that
\[ \|V_K - \tilde{V}_K T\|_F \to 0, \quad \text{as } N \to \infty. \]
(34)

Bounding $\|W_K R - W_K Q\|_F$. If we show that $\|X - \tilde{X}\| \to 0$ as $N \to \infty$, we can then mimic the procedure in [52] and [33] to show that the difference between the leading $K$ eigenvectors of $\mathcal{X}$ and $\tilde{X}$ also vanishes. Hence, we are left to show that $\|X - \tilde{X}\| \to 0$ as $N \to \infty$. From the definitions of $X$ and $\tilde{X}$, it follows that
\[ \|X - \tilde{X}\| \leq 0.5\|H^2 - \mathcal{H}^2\| + 1/2\|\arccos(C^{-1}\mathcal{H}^2C^{-1}) \odot \mathcal{H}^{-2} - \arccos(C^{-1}H^2C^{-1}) \odot H^2). \]
(35)

To bound the difference between the sampled and expected filters, we have that
\[ \|H^2 - \mathcal{H}^2\| = \left\| \left( \sum_{\ell=0}^{L} h_\ell \tilde{A}_\ell \right)^2 - \left( \sum_{\ell=0}^{L} h_\ell \tilde{A}_\ell \right)^2 \right\| \]
\[ = \left\| \sum_{\ell=0}^{2L} \alpha_\ell (\tilde{A}_\ell - \bar{A}_\ell)^2 \right\| \]
(36)

We can then leverage the fact that $\|\tilde{A}\| = \|\bar{A}\| = 1$ to see that $\|\tilde{A}_\ell - \bar{A}_\ell\| \leq \ell \|\tilde{A} - \bar{A}\|$. We thus get that
\[ \|H^2 - \mathcal{H}^2\| \leq \sum_{\ell=0}^{2L} \alpha_\ell \|\tilde{A} - \bar{A}\| \to 0, \quad \text{as } N \to \infty, \]
(37)

where the limiting behavior follows from [32]. Finally, to bound the second term in [35], we first note that the argument of the norm can be re-written as $\arccos(C^{-1}H^2C^{-1}) \odot (H^2 - \mathcal{H}^2)$ + $\arccos(C^{-1}\mathcal{H}^2C^{-1}) - \arccos(C^{-1}H^2C^{-1}) \odot \mathcal{H}^2$. The limit in [37] ensures that the first of these two terms vanishes. Similarly, it follows that $\|C^{-1}\mathcal{H}^2C^{-1} - C^{-1}H^2C^{-1}\| \to 0$ which, combined with the fact that $\arccos$ is a uniformly continuous function, we can always find an $N_\delta$ such that $\|\arccos(C^{-1}\mathcal{H}^2C^{-1}) - \arccos(C^{-1}H^2C^{-1})\| \leq \delta'$ with high probability. Combining this result with [37] and applying the Davis-Kahan Theorem as done to obtain [33] we get that
\[ \|W_K R - W_K Q\|_F \to 0, \quad \text{as } N \to \infty. \]
(38)

From replacing (31), (34), and (38) into (30) our result follows.

**APPENDIX C PROOF OF LEMMA [2]**

Recall that $\bar{A} = [\bar{A}]$, and define $\bar{\mathcal{H}} := \gamma I + (1 - \gamma)\bar{A}$ as the specific graph filter introduced in Section V-A using $\bar{A}$ as GSO. Let $X$ be given by equation (26), and denote by $\tilde{X}$ the expected squared Jacobian using the graph filter $\mathcal{H}$, i.e.,
\[ \tilde{X} = 0.5 \left( 11^T - \frac{1}{\pi} \arccos(\bar{c}^{-1}uU^TU^{-1}) \odot uu^T \right) \]
(39)

with $U = \bar{\mathcal{H}}P$ and where the matrix $\bar{c}$ is the counterpart of $C$ in (26), but using $U$ in lieu of $U$. Given the eigendecomposition $\bar{A} = V\Lambda V^T, \bar{X} = W\Sigma W^T, \tilde{A} = V\Lambda V^T$, and $\tilde{X} = W\Sigma W^T$, analogously to Lemma [1] we bound the difference between $V_K$ and $W_K$ by bounding the three terms in the right hand side of (30).

Bounding $\|V_K T - W_K R\|$. We have that $UU^T = \bar{\mathcal{H}}PP^T \mathcal{H}^T$. Since $P$ is a binary matrix indicating to which community belongs each node, $PP^T$ is a block diagonal matrix that captures the structure of the communities of the SBM. Then, because $\mathcal{H}$ is also block matrix with the same block pattern that the SBM, it turns out that the matrix $X$ is also a block matrix whose blocks coincide with the communities in the SBM graph. Therefore, the rest of the bound is analogous to that in Lemma [1].

Bounding $\|V_K - \tilde{V}_K T\|$. The relation between $A$ and $\bar{A}$ is the same as in Lemma [1] so the bound provided in (34) holds.

Bounding $\|W_K R - W_K Q\|$. To derive this bound we show that $\|uu^T - uu^T\| = \|PP^T \tilde{H}^T - PP^T \mathcal{H}^T\|$ goes to 0 as $N$ grows. From (37) we have that $\|H - \mathcal{H}\| \to 0$, as $N \to \infty$, and hence, $\|H - \mathcal{H}\| \to 0$, as $N \to \infty$. Therefore, it can be seen that
\[ \|UU^T - uu^T\| \to 0, \quad \text{as } N \to \infty. \]
(40)

With $\|UU^T - uu^T\|$ vanishing as $N$ grows, the remainder of the derivation of the bound is analogous to the one presented in [38].

**REFERENCES**

[1] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, “Graph signal processing: Overview, challenges, and applications.” *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[2] M. I. Jordan, Learning in graphical models. *Springer Science & Business Media*, 1998, vol. 89.

[3] E. D. Kolaczyk and G. Csárdi, Statistical analysis of network data with R. *Springer*, 2014, vol. 65.

[4] P. Djuric and C. Richard, *Cooperative and Graph Signal Processing: Principles and Applications*. Academic Press, 2018.
