arXiv:2003.08414v2 [cs.LG] 24 Jul 2020

Scattering GCN: Overcoming Oversmoothness in Graph Convolutional Networks

Yimeng Min[*]
Mila – Quebec AI Institute
Montreal, QC, Canada
minyimen@mila.quebec

Frederik Wenkel[*]
Dept. of Math. and Stat.
Université de Montréal
Mila – Quebec AI Institute
Montreal, QC, Canada
frederik.wenkel@umontreal.ca

Guy Wolf
Dept. of Math. and Stat.
Université de Montréal
Mila – Quebec AI Institute
Montreal, QC, Canada
guy.wolf@umontreal.ca

Abstract

Graph convolutional networks (GCNs) have shown promising results in processing graph data by extracting structure-aware features. This gave rise to extensive work in geometric deep learning, focusing on designing network architectures that ensure neuron activations conform to regularity patterns within the input graph. However, in most cases the graph structure is only accounted for by considering the similarity of activations between adjacent nodes, which limits the capabilities of such methods to discriminate between nodes in a graph. Here, we propose to augment conventional GCNs with geometric scattering transforms and residual convolutions. The former enables band-pass filtering of graph signals, thus alleviating the so-called oversmoothing often encountered in GCNs, while the latter is introduced to clear the resulting features of high-frequency noise. We establish the advantages of the presented Scattering GCN with both theoretical results establishing the complementary benefits of scattering and GCN features, as well as experimental results showing the benefits of our method compared to leading graph neural networks for semi-supervised node classification, including the recently proposed GAT network that typically alleviates oversmoothing using graph attention mechanisms.

1 Introduction

Deep learning approaches are at the forefront of modern machine learning. While they are effective in a multitude of applications, their most impressive results are typically achieved when processing data with inherent structure that can be used to inform the network architecture or the neuron connectivity design. For example, image processing tasks gave rise to convolutional neural networks that rely on spatial organization of pixels, while time series analysis gave rise to recurrent neural networks that leverage temporal organization in their information processing via feedback loops and memory mechanisms. The success of neural networks in such applications, traditionally associated with signal processing, has motivated the emergence of geometric deep learning, with the goal of generalizing the design of structure-aware network architectures from Euclidean spatiotemporal structures to a wide range of non-Euclidean geometries that often underlie modern data.

Geometric deep learning approaches typically use graphs as a model for data geometries, either by constructing them from input data (e.g., via similarity kernels) or directly given as quantified interactions between data points [1]. Using such models, recent works have shown that graph neural networks (GNNs) perform well in multiple application fields, including biology, chemistry and social networks [2–4]. It should be noted that most GNNs consider each graph together with given

[*]Equal contribution

Preprint. Under review.

node features, as a generalization of images or audio signals, and thus aim to compute whole-graph representations. These in turn, can be applied to graph classification, for example when each graph represents the molecular structure of proteins or enzymes classified by their chemical properties [5–7].

On the other hand, methods such as graph convolutional networks (GCNs) presented by [4] consider node-level tasks and in particular node classification. As explained in [4], such tasks are often considered in the context of semi-supervised learning, as typically only a small portion of nodes of the graph possesses labels. In these settings, the entire dataset is considered as one graph and the network is tasked with learning node representations that infer information from node features as well as the graph structure. However, most state-of-the-art approaches for incorporating graph structure information in neural network operations aim to enforce similarity between representations of adjacent (or neighboring) nodes, which essentially implements local smoothing of neuron activations over the graph [8]. While such smoothing operations may be sufficiently effective in whole-graph settings, they often cause degradation of results in node processing tasks due to oversmoothing [8, 9], as nodes become indistinguishable with deeper and increasingly complex network architectures. Graph attention networks [10] have shown promising results in overcoming such limitations by introducing adaptive weights for graph smoothing via message passing operations, using attention mechanisms computed from node features and masked by graph edges. However, these networks still essentially rely on enforcing similarity (albeit adaptive) between neighboring nodes, while also requiring more intricate training as their attention mechanism requires gradient computations driven not only by graph nodes, but also by graph edges. We refer the reader to the supplement for further discussion of related work and recent advances in node processing with GNNs.

In this paper, we propose a new approach for node-level processing in GNNs by introducing neural pathways that encode higher-order forms of regularity in graphs. Our construction is inspired by recently proposed geometric scattering networks [11–13], which have proven effective for whole-graph representation and classification. These networks generalize the Euclidean scattering transform, which was originally presented by [14] as a mathematical model for convolutional neural networks. In graph settings, the scattering construction leverages deep cascades of graph wavelets [15, 16] and pointwise nonlinearities to capture multiple modes of variation from node features or labels. Using the terminology of graph signal processing, these can be considered as generalized band-pass filtering operations, while GCNs (and many other GNNs) can be considered as relying on low-pass filters only. Our approach combines together the merits of GCNs on node-level tasks with those of scattering networks known from whole-graph tasks, by enabling learned node-level features to encode geometric information beyond smoothed activation signals, thus alleviating oversmoothing concerns often raised in GCN approaches. We discuss the benefits of our approach and demonstrate its advantages over GCNs and other popular graph processing approaches for semi-supervised node classification, including significant improvements on the DBLP graph dataset from [17].

Notations: We denote matrices and vectors with bold letters with uppercase letters representing matrices and lowercase letters representing vectors. In particular, $\boldsymbol{I}_n \in \mathbb{R}^{n\times n}$ is used for the identity matrix and $\mathbf{1}_n \in \mathbb{R}^n$ denotes the vector with ones in every component. We write $\langle ., . \rangle$ for the standard scalar product in $\mathbb{R}^n$. We will interchangeably consider functions of graph nodes as vectors indexed by the nodes, implicitly assuming a correspondence between a node and a specific index. This carries over to matrices, where we relate nodes to column or row indices. We further use the abbreviation $[n] \coloneqq \{1, \dots, n\}$ where $n \in \mathbb{N}$ and write $\mathbb{N}_0 \coloneqq \mathbb{N} \cup \{0\}$.

2 Graph Signal Processing

Let $G = (V, E, w)$ be a weighted graph with $V \coloneqq \{v_1, \dots, v_n\}$ the set of nodes, $E \subset \{\{v_i, v_j\} \in V \times V, i \neq j\}$ the set of (undirected) edges and $w : E \to (0, \infty)$ assigning (positive) edge weights to the graph edges. We note that w can equivalently be considered as a function of $V \times V$, where we set the weights of non-adjacent node pairs to zero. We define a *graph signal* as a function $x : V \to \mathbb{R}$ on the nodes of G and aggregate them in a signal vector $\boldsymbol{x} \in \mathbb{R}^n$ with the i^{th} entry being $x(v_i)$.

We define the (combinatorial) *graph Laplacian* matrix $\boldsymbol{L} \coloneqq \boldsymbol{D} - \boldsymbol{W}$, where $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ is the *weighted adjacency matrix* of the graph G given by

$$\boldsymbol{W}[v_i, v_j] \coloneqq \begin{cases} w(v_i, v_j) & \text{if } \{v_i, v_j\} \in E \\ 0 & \text{otherwise} \end{cases},$$

and $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ is the *degree matrix* of G defined by $\boldsymbol{D} \coloneqq \mathrm{diag}(d_1, \ldots, d_n)$ with $d_i \coloneqq \deg(v_i) \coloneqq \sum_{j=1}^{n} \boldsymbol{W}[v_i, v_j]$ being the *degree* of the node v_i. In practice, we work with the (symmetric) *normalized Laplacian* matrix $\boldsymbol{\mathcal{L}} \coloneqq \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2} = \boldsymbol{I}_n - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$. It can be verified that $\boldsymbol{\mathcal{L}}$ is symmetric and positive semi-definite and can thus be orthogonally diagonalized as $\boldsymbol{\mathcal{L}} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T = \sum_{i=1}^{n} \lambda_i \boldsymbol{q}_i \boldsymbol{q}_i^T$, where $\boldsymbol{\Lambda} \coloneqq \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix with the eigenvalues on the main diagonal and $\boldsymbol{Q}$ is an orthogonal matrix containing the corresponding normalized eigenvectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n \in \mathbb{R}^n$ as its columns.

A detailed study (see, e.g., [18]) of the eigenvalues reveals that $0 = \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n \leqslant 2$. We can interpret the $\lambda_i, i \in [n]$ as the frequency magnitudes and the $\boldsymbol{q}_i$ as the corresponding Fourier modes. We accordingly define the *Fourier transform* of a signal vector $\boldsymbol{x} \in \mathbb{R}^n$ by $\hat{\boldsymbol{x}}[i] = \langle \boldsymbol{x}, \boldsymbol{q}_i \rangle$ for $i \in [n]$. The corresponding inverse Fourier transform is given by $\boldsymbol{x} = \sum_{i=1}^{n} \hat{\boldsymbol{x}}[i]\boldsymbol{q}_i$. Note that this can be written compactly as $\hat{\boldsymbol{x}} = \boldsymbol{Q}^T\boldsymbol{x}$ and $\boldsymbol{x} = \boldsymbol{Q}\hat{\boldsymbol{x}}$. Finally, we introduce the concept of *graph convolutions*. We define a filter $g : V \to \mathbb{R}$ defined on the set of nodes and want to convolve the corresponding filter vector $\boldsymbol{g} \in \mathbb{R}^n$ with a signal vector $\boldsymbol{x} \in \mathbb{R}^n$, i.e. $\boldsymbol{g} \star \boldsymbol{x}$. To explicitly compute this convolution, we recall that in the Euclidean setting, the convolution of two signals equals the product of their corresponding frequencies. This property generalizes to graphs [19] in the sense that $\widehat{(\boldsymbol{g} \star \boldsymbol{x})}[i] = \hat{\boldsymbol{g}}[i]\hat{\boldsymbol{x}}[i]$ for $i \in [n]$. Applying the inverse Fourier transform yields

$$\boldsymbol{g} \star \boldsymbol{x} = \sum_{i=1}^{n} \hat{\boldsymbol{g}}[i]\hat{\boldsymbol{x}}[i]\boldsymbol{q}_i = \sum_{i=1}^{n} \hat{\boldsymbol{g}}[i]\langle \boldsymbol{q}_i, \boldsymbol{x} \rangle \boldsymbol{q}_i = \boldsymbol{Q}\widehat{\boldsymbol{G}}\boldsymbol{Q}^T\boldsymbol{x},$$

where $\widehat{\boldsymbol{G}} \coloneqq \mathrm{diag}(\hat{\boldsymbol{g}}) = \mathrm{diag}(\hat{\boldsymbol{g}}[1], \ldots, \hat{\boldsymbol{g}}[n])$. Hence, convolutional graph filters can be parameterized by considering the Fourier coefficients in $\widehat{\boldsymbol{G}}$.

Furthermore, it can be verified [20] that when these coefficients are defined as polynomials $\hat{\boldsymbol{g}}[i] \coloneqq \sum_k \gamma_k \lambda_i^k$ for $i \in \mathbb{N}$ of the Laplacian eigenvalues in $\boldsymbol{\Lambda}$ (i.e. $\widehat{\boldsymbol{G}} = \sum_k \gamma_k \boldsymbol{\Lambda}^k$), the resulting filter convolution are localized in space and can be written in terms of $\mathcal{L}$ as $\boldsymbol{g} \star \boldsymbol{x} = \sum_k \gamma_k \mathcal{L}^k \boldsymbol{x}$ without requiring spectral decomposition of the normalized Laplacian. This motivates the standard practice [4, 20–22] of using filters that have polynomial forms, which we follow here as well.

3 Graph Convolutional Network

Graph convolutional networks (GCNs), introduced in [4], consider semi-supervised settings where only a small potion of the nodes is labeled. They leverage intrinsic geometric information encoded in the adjacency matrix $\boldsymbol{W}$ together with node labels by constructing a convolutional filter parametrized by $\hat{\boldsymbol{g}}[i] \coloneqq \theta(2 - \lambda_i)$, where the choice of a single learnable parameter is made to avoid overfitting. This parametrization yields a convolutional filtering operation given by

$$\boldsymbol{g}_\theta \star \boldsymbol{x} = \theta \left(\boldsymbol{I}_n + \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2} \right) \boldsymbol{x}. \tag{1}$$

The matrix $\boldsymbol{I}_n + \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$ has eigenvalues in $[0, 2]$. This could lead to vanishing or exploding gradients. This issue is addressed by the following renormalization trick [4]: $\boldsymbol{I}_n + \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2} \to \tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{W}}\tilde{\boldsymbol{D}}^{-1/2}$, where $\tilde{\boldsymbol{W}} \coloneqq \boldsymbol{I}_n + \boldsymbol{W}$ and $\tilde{\boldsymbol{D}}$ a diagonal matrix with $\tilde{\boldsymbol{D}}[v_i, v_i] \coloneqq \sum_{j=1}^{n} \tilde{\boldsymbol{W}}[v_i, v_j]$ for $i \in [n]$. This operation replaces the features of the nodes by a weighted average of itself and its neighbors. Note that the repeated execution of graph convolutions will enforce similarity throughout higher-order neighborhoods with order equal to the number of stacked layers. Setting $\boldsymbol{A} \coloneqq \tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{W}}\tilde{\boldsymbol{D}}^{-1/2}$, the complete layer-wise propagation rule takes the form $\boldsymbol{h}_j^\ell = \sigma\big(\sum_{i=1}^{N_{\ell-1}} \theta_i^\ell \boldsymbol{A}\boldsymbol{h}_i^{\ell-1}\big)$. Here, ℓ indicates the layer with N_ℓ neurons, $\boldsymbol{h}_j^\ell \in \mathbb{R}^n$ the activation vector of the j^{th} neuron, θ_i^ℓ the learned parameter of the convolution with the i^{th} incoming activation vector from the preceding layer and $\sigma(.)$ an element-wise applied activation function. Written in matrix notation, this gives

$$\boldsymbol{H}^\ell = \sigma\left(\boldsymbol{A}\boldsymbol{H}^{\ell-1}\boldsymbol{\Theta}^\ell \right), \tag{2}$$

where $\boldsymbol{\Theta}^\ell \in \mathbb{R}^{N_{\ell-1}\times N_\ell}$ is the weight-matrix of the ℓ^{th} layer and $\boldsymbol{H}^\ell \in \mathbb{R}^{n\times N_\ell}$ contains the activations outputted by the ℓ^{th} layer.

We remark that the above explained GCN model can be interpreted as a low-pass operation. For the sake of simplicity, let us consider the convolutional operation (Eq. 1) before the reparametrization trick. If we observe the convolution operation as the summation $\boldsymbol{g}_\theta \star \boldsymbol{x} = \sum_{i=1}^{n} \boldsymbol{\gamma}_i \hat{\boldsymbol{x}}[i] \boldsymbol{q}_i$, we clearly see that higher weights $\boldsymbol{\gamma}_i = \theta(2-\lambda_i)$ are put on the low-frequency harmonics, while high-frequency harmonics are progressively less involved as $0 = \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n \leqslant 2$. This indicates that the model can only access a diminishing portion of the original information contained in the input signal the more graph convolutions are stacked. This observation is in line with the well-known oversmoothing problem [8] related to GCN models. The repeated application of graph convolutions will successively smooth the signals of the graph such that nodes cannot be distinguished anymore.

4 Geometric Scattering

In this section, we recall the construction of geometric scattering on graphs. This construction is based on the *lazy random walk* matrix

$$\boldsymbol{P} \coloneqq \frac{1}{2}\big(\boldsymbol{I}_n + \boldsymbol{W}\boldsymbol{D}^{-1}\big),$$

which is closely related to the *graph random walk* defined as a Markov process with transition matrix $\boldsymbol{R} \coloneqq \boldsymbol{W}\boldsymbol{D}^{-1}$. The matrix $\boldsymbol{P}$ however allows self loops while normalizing by a factor of two in order to retain a Markov process. Therefore, considering a distribution $\boldsymbol{\mu}_0 \in \mathbb{R}^n$ of the initial position of the lazy random walk, its positional distribution after t steps is encoded by $\boldsymbol{\mu}_t = \boldsymbol{P}^t \boldsymbol{\mu}_0$.

As discussed in [12], the propagation of a graph signal vector $\boldsymbol{x} \in \mathbb{R}^n$ by $\boldsymbol{x}_t = \boldsymbol{P}^t \boldsymbol{x}$ performs a low-pass operation that preserves the zero-frequencies of the signal while suppressing high frequencies. In geometric scattering, this low-pass information is augmented by introducing the *wavelet* matrices $\boldsymbol{\Psi}_k \in \mathbb{R}^{n \times n}$ of scale 2^k, $k \in \mathbb{N}_0$,

$$\begin{cases} \boldsymbol{\Psi}_0 \coloneqq \boldsymbol{I}_n - \boldsymbol{P}, \\ \boldsymbol{\Psi}_k \coloneqq \boldsymbol{P}^{2^{k-1}} - \boldsymbol{P}^{2^k} = \boldsymbol{P}^{2^{k-1}}\big(\boldsymbol{I}_n - \boldsymbol{P}^{2^{k-1}}\big), \quad k \geq 1. \end{cases} \tag{3}$$

This leverages the fact that high frequencies can be recovered with multiscale wavelet transforms, e.g., by decomposing nonzero frequencies into dyadic frequency bands. The operation $(\boldsymbol{\Psi}_k \boldsymbol{x})[v_i]$ collects signals from a neighborhood of order 2^k, but extracts multiscale differences rather than averaging over them. The wavelets in Eq. 3 can be organized in a filter bank $\{\boldsymbol{\Psi}_k, \tilde{\boldsymbol{\Psi}}_K\}_{0 \leq k \leq K}$, where $\tilde{\boldsymbol{\Psi}}_K \coloneqq \boldsymbol{P}^{2^K}$ is a pure low-pass filter. The telescoping sum of the matrices in this filter bank constitutes the identity matrix, thus enabling to reconstruct processed signals from their filter responses. Further studies of this construction and its properties (e.g., energy preservation) appear in [23] and related work.

Geometric scattering was originally introduced in the context of whole-graph classification and consisted of aggregating *scattering features*. These are stacked wavelet transforms parameterized via tuples $J \coloneqq (k_1, \ldots, k_m) \in \cup_{m \in \mathbb{N}} \mathbb{N}_0^m$ containing the bandwidth scale parameters, which are separated by element-wise absolute value nonlinearities according to

$$\boldsymbol{\Phi}_J \boldsymbol{x} \coloneqq \boldsymbol{\Psi}_{k_m} |\boldsymbol{\Psi}_{k_{m-1}} \ldots |\boldsymbol{\Psi}_{k_2} |\boldsymbol{\Psi}_{k_1} \boldsymbol{x}| | \ldots |,$$

where m corresponds to the length of the tuple J. The scattering features are aggregated over the whole graph by taking q^{th}-order moments over the set of nodes,

$$S(\boldsymbol{x}, J, q) \coloneqq \sum_{i=1}^{n} |\boldsymbol{\Phi}_J \boldsymbol{x}[v_i]|^q. \tag{4}$$

As our work is devoted to the study of node-based classification, we reinvent this approach in a new context, keeping the scattering transforms $\boldsymbol{\Phi}_J$ on a node-level by dismissing the aggregation step in Eq. 4. For each tuple J, we define the following scattering propagation rule, which mirrors the GCN rule but replaces the low-pass filter by a geometric scattering operation resulting in

$$\boldsymbol{H}^\ell = \sigma\left(\boldsymbol{\Phi}_J \boldsymbol{H}^{\ell-1} \boldsymbol{\Theta}^\ell\right). \tag{5}$$

We note that in practice, we only choose a subset of tuples, which is chosen as part of the network design explained in the following section.

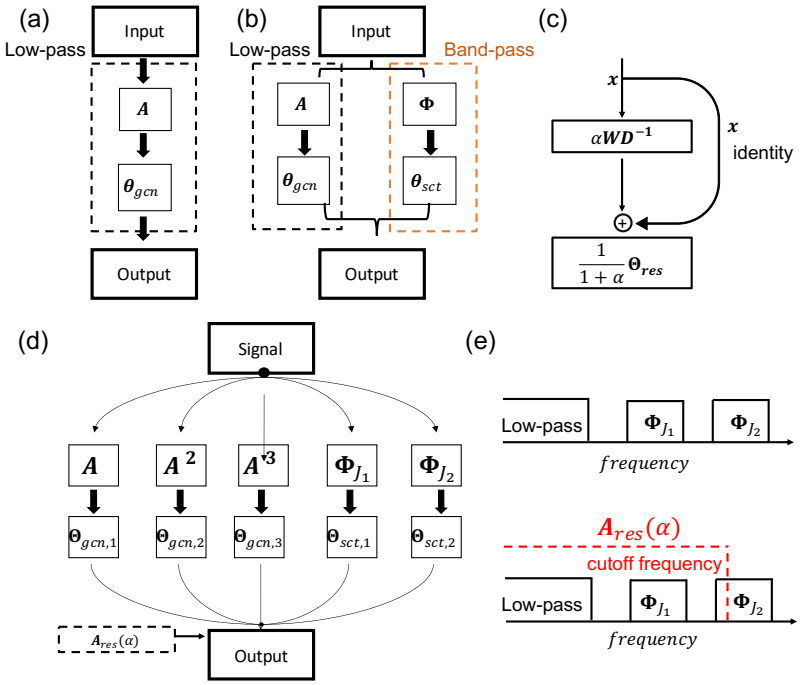


Figure 1: (a,b) Comparison between GCN and our network: we add band-pass channels to collect different frequency components; (c) Graph residual convolution layer; (d) Band-pass layers; (e) Schematic depiction in the frequency domain.

5 Combining GCN and Scattering Models

To combine the benefits of GCN models and geometric scattering adapted to the node level, we now propose a hybrid network architecture as shown in Fig. 1. It combines low-pass operations based on GCN models with band-pass operations based on geometric scattering. To define the layer-wise propagation rule, we introduce

$$\boldsymbol{H}_{gcn}^{\ell} \coloneqq \left[\boldsymbol{H}_{gcn,1}^{\ell} \,\|\, \ldots \,\|\, \boldsymbol{H}_{gcn,C_{gcn}}^{\ell}\right] \quad \text{and} \quad \boldsymbol{H}_{sct}^{\ell} \coloneqq \left[\boldsymbol{H}_{sct,1}^{\ell} \,\|\, \ldots \,\|\, \boldsymbol{H}_{sct,C_{sct}}^{\ell}\right],$$

which are the concatenations of channels $\left\{\boldsymbol{H}_{gcn,k}^{\ell}\right\}_{k=1}^{C_{gcn}}$ and $\left\{\boldsymbol{H}_{sct,k}^{\ell}\right\}_{k=1}^{C_{sct}}$, respectively. Every $\boldsymbol{H}_{gcn,k}^{\ell}$ is defined according to Eq. 2 with the slight modification of added biases and powers of $\boldsymbol{A}$,

$$\boldsymbol{H}_{gcn,k}^{\ell} \coloneqq \sigma\left(\boldsymbol{A}^k \boldsymbol{H}^{\ell-1} \boldsymbol{\Theta}_{gcn,k}^{\ell} + \boldsymbol{B}_{gcn,k}^{\ell}\right).$$

Note that every GCN filter uses a different propagation matrix $\boldsymbol{A}^k$ and therefore aggregates information from k-step neighborhoods. Similarly, we proceed with $\boldsymbol{H}_{sct,k}^{\ell}$ according to Eq. 5 and calculate

$$\boldsymbol{H}_{sct,k}^{\ell} \coloneqq \sigma\left(\boldsymbol{\Phi}_{J_k} \boldsymbol{H}^{\ell-1} \boldsymbol{\Theta}_{sct,k}^{\ell} + \boldsymbol{B}_{sct,k}^{\ell}\right),$$

where $J_k \in \bigcup_{m\in\mathbb{N}} \mathbb{N}_0^m$, $k = 1, \ldots, C_{sct}$ enables scatterings of different orders and scales. Finally, the GCN components and scattering components get concatenated to

$$\boldsymbol{H}^{\ell} \coloneqq \left[\boldsymbol{H}_{gcn}^{\ell} \,\|\, \boldsymbol{H}_{sct}^{\ell}\right]. \tag{6}$$

The learned parameters are the weight matrices $\boldsymbol{\Theta}_{gcn,k}^{\ell}, \boldsymbol{\Theta}_{sct,k}^{\ell} \in \mathbb{R}^{N_{\ell-1}\times N_{\ell}}$ coming from the convolutional and scattering layers. These are complemented by vectors of the biases $\boldsymbol{b}_{gcn,k}^{\ell}, \boldsymbol{b}_{sct,k}^{\ell} \in \mathbb{R}^{N_{\ell}}$, which are transposed and vertically concatenated n times to the matrices $\boldsymbol{B}_{gcn,k}, \boldsymbol{B}_{sct,k} \in$

$\mathbb{R}^{n \times N_\ell}$. To simplify notation, we assume here that all channels use the same number of neurons (N_ℓ). Waiving this assumption would slightly complicate the notation but works perfectly fine in practice.

In this work, for simplicity, and because it is sufficient to establish our claim, we limit our architecture to three GCN channels and two scattering channels as illustrated in Fig. 1 (b). Inspired by the aggregation step in classical geometric scattering, we use $\sigma(.) := |\ .\ |^q$ as our nonlinearity. However, unlike the powers in Eq. 4, the q^{th} power is applied at the node-level here instead of being aggregated as moments over the entire graph, thus retaining the distinction between node-wise activations.

We set the input of the first layer $\boldsymbol{H}^0$ to have the original node features as the graph signal. Each subchannel (GCN or scattering) transforms the original feature space to a new hidden space with the dimension determined by the number of neurons encoded in the columns of the corresponding submatrix of $\boldsymbol{H}^\ell$. These transformations are learned by the network via the weights and biases. Larger matrices $\boldsymbol{H}^\ell$ (i.e., more columns as the number of nodes in the graph is fixed) indicate that the weight matrices have more parameters to learn. Thus, the information in these channels can be propagated well and will be sufficiently represented.

In general, the width of a channel is relevant for the importance of the captured regularities. A wider channel suggests that these frequency components are more critical and need to be sufficiently learned. Reducing the width of the channel suppresses the magnitude of information that can be learned from a particular frequency window. For more details and analysis of specific design choices in our architecture we refer the reader to the ablation study provided in the supplement.

6 Graph Residual Convolution

Using the combination of GCN and scattering architectures, we collect multiscale information at the node level. This information is aggregated from different localized neighborhoods, which may exhibit vastly different frequency spectra. This comes for example from varying label rates in different graph substructures. In particular, very sparse graph sections can cause problems when the scattering features actually learn the difference between labeled and unlabeled nodes, creating high-frequency noise. In the classical geometric scattering used for whole-graph representation, geometric moments were used to aggregate the node-based information, serving at the same time as a low-pass filter. As we want to keep the information localized on the node level, we choose a different approach inspired by skip connections in residual neural networks [24]. Conceptually, this low-pass filter, which we call *graph residual convolution*, reduces the captured frequency spectrum up to a cutoff frequency as depicted in Fig. 1 (e).

The graph residual convolution matrix, governed by the hyperparameter α, is given by $\boldsymbol{A}_{res}(\alpha) = \frac{1}{\alpha+1}(\boldsymbol{I}_n + \alpha \boldsymbol{W}\boldsymbol{D}^{-1})$ and we apply it after the hybrid layer of GCN and scattering filters. For $\alpha = 0$ we get the identity (no cutoff), while $\alpha \to \infty$ results in $\boldsymbol{R} = \boldsymbol{W}\boldsymbol{D}^{-1}$. This can be interpreted as an interpolation between the completely lazy random walk and the non-resting random walk $\boldsymbol{R}$. We apply the graph residual layer on the output $\boldsymbol{H}^\ell$ of the Scattering GCN layer (Eq. 6). The update rule for this step, illustrated in Fig. 1 (c), is then expressed by $\boldsymbol{H}^{\ell+1} = \boldsymbol{A}_{res}(\alpha)\boldsymbol{H}^\ell\boldsymbol{\Theta}_{res} + \boldsymbol{B}_{res}$, where $\boldsymbol{\Theta}_{res} \in \mathbb{R}^{N \times N_{\ell+1}}$ are learned weights, $\boldsymbol{B}_{res} \in \mathbb{R}^{n \times N_{\ell+1}}$ are learned biases (similar to the notations used previously), and N is the number of features of the concatenated layer $\boldsymbol{H}^\ell$. If $\boldsymbol{H}^{\ell+1}$ is the final layer, we choose $N_{\ell+1}$ equal to the number of classes.

7 Additional Information Introduced by Node-level Scattering Features

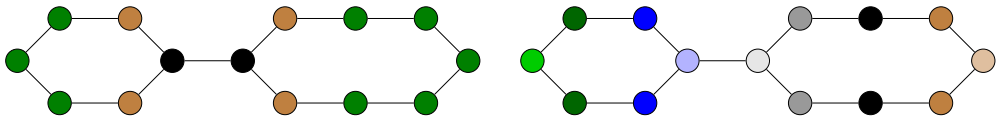
Before empirically verifying the viability of the proposed architecture in node classification tasks, we first discuss and demonstrate the additional information provided by scattering channels beyond that provided by traditional GCN channels. We first consider information carried by node features, treated as graph signals, and in particular their regularity over the graph. As discussed in Sec. 3, such regularity is traditionally considered only via smoothness of signals over the graph, as only low frequencies are retained by (local) smoothing operations. Band-pass filtering, on the other hand, can retain other forms of regularity such as periodic or harmonic patterns. The following lemma demonstrates this difference between GCN and scattering channels.

Lemma 1. *Consider a cyclic graph on $2n$ nodes, $n \in \mathbb{N}$, and let $\boldsymbol{x} \in \mathbb{R}^{2n}$ be a 2-periodic signal on it (i.e., $\boldsymbol{x}_{2\ell-1} = a$ and $\boldsymbol{x}_{2\ell} = b$, for $\ell \in [n]$ for some $a \neq b \in \mathbb{R}$). Then, for any $\theta \in \mathbb{R}$, the GCN filtering $\boldsymbol{g}_\theta \star \boldsymbol{x}$ from Eq. 1 yields a constant signal, while the scattering filter $\boldsymbol{\Psi}_0 \boldsymbol{x}$ from Eq. 3 still produces a 2-periodic signal. Further, this result extends to any finite linear cascade of such filters (i.e., $\boldsymbol{g}_\theta \star \cdots \star \boldsymbol{g}_\theta \star \boldsymbol{x}$ or $\boldsymbol{\Psi}_0 \cdots \boldsymbol{\Psi}_0 \boldsymbol{x}$ with $k \in \mathbb{N}$ filter applications in each).*

While this is only a simple example, it already indicates a fundamental difference between the regularity patterns considered in graph convolutions compared to our approach. Indeed, it implies that if a smoothing convolutional filter encounters alternating signals on isolated cyclic substructures within a graph, their node features become indistinguishable, while scattering channels (with appropriate scales, weights and bias terms) will be able to make this distinction. Moreover, this difference can be generalized further beyond cyclic structures to consider features encoding two-coloring information on constant-degree bipartite graphs, as shown in the following lemma. We refer the reader to the supplement for a proof of this lemma, which also covers the previous one as a particular case.

Lemma 2. *Consider a bipartite graph on $n \in \mathbb{N}$ nodes with constant node degree β. Let $\boldsymbol{x} \in \mathbb{R}^n$ be a 2-coloring signal (i.e., with one part assigned constant a and the other b, for some $a \neq b \in \mathbb{R}$). Then, for any $\theta \in \mathbb{R}$, the GCN filtering $\boldsymbol{g}_\theta \star \boldsymbol{x}$ from Eq. 1 yields a constant signal, while the scattering filter $\boldsymbol{\Psi}_0 \boldsymbol{x}$ from Eq. 3 still produces a (non-constant) 2-coloring of the graph. Further, this result extends to any finite linear cascade of such filters (i.e., $\boldsymbol{g}_\theta \star \cdots \star \boldsymbol{g}_\theta \star \boldsymbol{x}$ or $\boldsymbol{\Psi}_0 \cdots \boldsymbol{\Psi}_0 \boldsymbol{x}$ with $k \in \mathbb{N}$ filter applications in each).*

Beyond the information encoded in node features, graph wavelets encode geometric information even when it is not carried by input signals. Such a property has already been established, e.g., in the context of community detection, where white noise signals can be used in conjunction with graph wavelets to cluster nodes and reveal faithful community structures [25]. To demonstrate a similar property in the context of GCN and scattering channels, we give an example of a simple graph structure with two cyclic substructures of different sizes (or cycle lengths) that are connected by one bottleneck edge. In this case, it can be verified that even with constant input signals, some geometric information is encoded by its convolution with graph filters as illustrated in Fig. 2 (exact calculations are shown in Sec. B of the supplement). However, as demonstrated in this case, while the information provided by the GCN filter responses $\boldsymbol{g}_\theta \star \boldsymbol{x}$ from Eq. 1 is not constant, it does not distinguish between the two cyclic structures (and a similar pattern can be verified for $\boldsymbol{A}\boldsymbol{x}$). Formally, each node u in one cycle is shown to have at least one node v in the other with the same filter response (i.e., $\boldsymbol{g}_\theta \star \boldsymbol{x}(u) = \boldsymbol{g}_\theta \star \boldsymbol{x}(v)$). In contrast, the information extracted by the wavelet filter response $\boldsymbol{\Psi}_3 \boldsymbol{x}$ (used in geometric scattering) distinguishes between cycles and would allow for their separation. We note that this property generalizes to other cycle lengths as discussed in the supplement, but leave more extensive study of geometric information encoding in graph wavelets to future work.



(a) Filter response of $\boldsymbol{g}_\theta \star \boldsymbol{x}$ (b) Filter response of $\boldsymbol{\Psi}_3 \boldsymbol{x}$

Figure 2: Filter responses for (a) the GCN filter (Eq. 1) and (b) a scattering filter applied to a constant signal $\boldsymbol{x}$ over a graph with two cyclic substuctures connected by a single-edge bottleneck. Color coding differs slightly between plots, but is consistent within each plot, indicating nodes with numerically indistinguishable response values.

8 Empirical Results

To evaluate our Scattering GCN approach, we compare it to several established methods for semi-supervised node classification, including the original GCN [4], which is known to be subject to the oversmoothing problem, as discussed in [8], and Sec. 1 and 3 here. Further, we compare our approach with two recent methods that address the oversmoothing problem. The approach in [8] directly addresses oversmoothing in GCNs by using partially absorbing random walks [26] to mitigate rapid mixing of node features in highly connected graph regions. The graph attention network (GAT) [10] indirectly addresses oversmoothing by training adaptive node-wise weighting of the smoothing

Table 1: Dataset characteristics: number of nodes, edges, and features; mean $\pm$ std. of node degrees; ratio of #edges to #nodes.

Dataset	Nodes	Edges	Features	Degrees	$\frac{\text{Edges}}{\text{Nodes}}$
Citeseer	3,327	4,732	3,703	3.77$\pm$3.38	1.42
Cora	2,708	5,429	1,433	4.90$\pm$5.22	2.00
Pubmed	19,717	44,338	500	5.50$\pm$7.43	2.25
DBLP	17,716	105,734	1639	6.97$\pm$9.35	5.97

Table 2: Classification accuracy (top two marked in bold; best one underlined) of Scattering GCN on four benchmark datasets compared to four other GNNs [10, 8, 4, 20], a non-GNN approach [27] based on belief propagation, and a baseline using only node features with linear SVM.

Model	Citeseer	Cora	Pubmed	DBLP
Scattering GCN (ours)	**71.7**	**<u>83.6</u>**	**<u>79.4</u>**	**<u>81.5</u>**
GAT [10]	**<u>72.5</u>**	**83.0**	79.0	**66.1**
Partially absorbing [8]	71.2	81.7	**79.2**	56.9
GCN [4]	70.3	81.5	79.0	59.3
Chebyshev [20]	69.8	78.1	74.4	57.3
Label Propagation [27]	58.2	77.3	71.0	53.0
Node features (SVM)	61.1	58.0	49.9	48.2

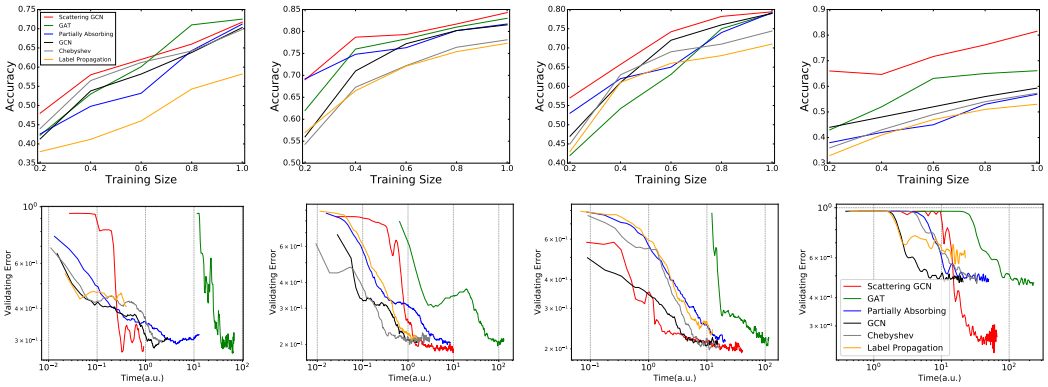
operation via an attention mechanism. Furthermore, we also include two alternatives to GCN networks based on Chebyshev polynomial filters [20] and belief propagation of label information [27] computed via Gaussian random fields. Finally, to verify the contribution of incorporated geometric (graph structure) information in the presented benchmarks, we add a baseline SVM classifier, acting directly on the node features, without considering at all the graph edges.

The methods [4, 8, 10, 20, 27] were all executed using the original implementations accompanying their publications. These are tuned and evaluated using the standard splits provided for the benchmark datasets for fair comparison. We ensure that the reported classification accuracies agree with previously published results when available. The tuning of our method (including hyperparameters and composition of GCN and scattering channels) on each dataset was done via grid search (over a fixed set of choices for all datasets) using the same cross validation setup used to tune competing methods. For further details, we refer the reader to the supplement, which contains an ablation study evaluating the importance of each component in our proposed architecture.

Our comparisons are based on four popular graph datasets with varying sizes and connectivity structures summarized in Tab. 1 (see, e.g., [28] for Citeseer, Cora, and Pubmed, and [17] for DBLP). We order the datasets by increasing connectivity structure, reflected by their node degrees and edges-to-nodes ratios. As discussed in [8], increased connectivity leads to faster mixing of node features in GCN, exacerbating the oversmoothing problem (as nodes quickly become indistinguishable) and degrading classification performance. Therefore, we expect the impact of scattering channels and the relative improvement achieved by Scattering GCN to correspond to the increasing connectivity order of datasets in Tab. 1, which is maintained for our reported results in Tab. 2 and Fig. 3.

We first consider test classification accuracy reported in Tab. 2, which shows that our approach outperforms other methods on three out of the four considered datasets. On the remaining one (namely Citeseer) we are only outperformed by GAT. However, we note that this dataset has the weakest connectivity structure (see Tab. 1) and the most informative node features (e.g., achieving 61.1% accuracy via linear SVM without considering any graph information). In contrast, on DBLP, which has the richest connectivity structure and least informative features (only 48.2% SVM accuracy), we significantly outperform GAT (over 15% improvement), which itself significantly outperforms all other methods (by 6.8% or more).

Next, we consider the impact of training size on classification performance as we are interested in semi-supervised settings where only a small portion of nodes in the graph are labelled. Fig. 3 (top) presents the classification accuracy (on validation set) for the training size reduced to 20%, 40%,



(a) Citeseer (b) Cora (c) Pubmed (d) DBLP

Figure 3: Impact of training set size (top) and training time (bottom) on classification accuracy and error (correspondingly); training size measured relative to the original training size of each dataset; training time and validation error plotted in logarithmic scale; runtime measured for all methods on the same hardware, using original implementations accompanying their publications.

60%, 80% and 100% of the original training size available for each dataset. These results indicate that Scattering GCN generally exhibits greater stability to sparse training conditions compared to other methods. Importantly, we note that on Citeseer, while GAT outperforms our method for the original training size, its performance degrades rapidly when training size is reduced below 60% of the original one, at which point Scattering GCN outperforms all other methods. We also note that on Pubmed, even a small decrease in training size (e.g., 80% of original) creates a significant performance gap between Scattering GCN and GAT, which we believe is due to node features being less independently informative in this case (see baseline in Tab. 2) compared to Citeseer and Cora.

Finally, in Fig. 3 (bottom), we consider the evolution of (validation) classification error during the training process. Overall, our results indicate that the training of Scattering GCN reaches low validation errors significantly faster than Partially Absorbing and GAT[2], which are the two other leading methods (in terms of final test accuracy in Tab. 2). On Pubmed, which is the largest dataset considered here (by number of nodes), our error decays at a similar rate to that of GCN, showing a notable gap over all other methods. On DBLP, which has a similar number of nodes but significantly more edges, Scattering GCN takes longer to converge (compared to GCN), but as discussed before, it also demonstrates a significant (double-digit) performance lead compared to all other methods.

9 Conclusion

Our study of semi-supervised node-level classification tasks for graphs presents a new approach to address some of the main concerns and limitations of GCN models. We discuss and consider richer notions of regularity on graphs to expand the GCN approach, which solely relies on enforcing smoothness over graph neighborhoods. This is achieved by incorporating multiple frequency bands of graph signals, which are typically not leveraged in traditional GCN models. Our construction is inspired by geometric scattering, which has mainly been used for whole-graph classification so far. Our results demonstrate several benefits of incorporating the elements presented here (i.e., scattering channels and residual convolutions) into GCN architectures. Furthermore, we expect the incorporation of these elements together in more intricate architectures to provide new capabilities of pattern recognition and local information extraction in graphs. For example, attention mechanisms could be used to adaptively tune scattering configurations at the resolution of each node, rather than the global graph level used here. We leave the exploration of such research avenues for future work.

[2]The horizontal shift shown for GAT in Fig. 3 (bottom), indicating increased training runtime (based on the original implementation accompanying [10]), could be explained by its optimization process requiring more weights than other methods and an intensive gradient computations driven not only graph nodes, but also by graph edges considered in the multihead attention mechanism.

Acknowledgements

The authors would like to thank Dongmian Zou for fruitful discussions. This work was partially funded by IVADO (l'institut de valorisation des données) and NIH grant R01GM135929. The content provided here is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies.

References

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

[2] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR, 2017.

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

[4] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International conference on learning representations*, 2016.

[5] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6533–6542, 2017.

[6] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[7] Boris Knyazev, Xiao Lin, Mohamed R Amer, and Graham W Taylor. Spectral multigraph networks for discovering and fusing relationships in molecules. *arXiv preprint arXiv:1811.09595*, 2018.

[8] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 3538–3545. Association for the Advancement of Artificial Intelligence, 2018.

[9] Hoang NT and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

[10] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[11] Fernando Gama, Alejandro Ribeiro, and Joan Bruna. Diffusion scattering transforms on graphs. In *International Conference on Learning Representations*, 2019.

[12] Feng Gao, Guy Wolf, and Matthew Hirn. Geometric scattering for graph data analysis. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 2122–2131, 2019.

[13] Dongmian Zou and Gilad Lerman. Graph convolutional neural networks via scattering. *Applied and Computational Harmonic Analysis*, 2019.

[14] Stéphane Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

[15] David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129 – 150, 2011.

[16] Ronald R. Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53 – 94, 2006.

[17] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. Tri-party deep network representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1895–1901, 2016.

[18] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[19] David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291, 2016.

[20] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3844–3852, 2016.

[21] Ana Susnjara, Nathanael Perraudin, Daniel Kressner, and Pierre Vandergheynst. Accelerated filtering on graphs using lanczos method. *arXiv preprint arXiv:1509.04537*, 2015.

[22] Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard S Zemel. Lanczosnet: Multi-scale deep graph convolutional networks. *arXiv preprint arXiv:1901.01484*, 2019.

[23] Michael Perlmutter, Feng Gao, Guy Wolf, and Matthew Hirn. Understanding graph neural networks with asymmetric geometric scattering transforms. *arXiv preprint arXiv:1911.06253*, 2019.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[25] T Mitchell Roddenberry, Michael T Schaub, Hoi-To Wai, and Santiago Segarra. Exact blind community detection from signals on multiple graphs. *arXiv preprint arXiv:2001.10944*, 2020.

[26] Xiao-Ming Wu, Zhenguo Li, Anthony Man-Cho So, John Wright, and Shih-Fu Chang. Learning with partially absorbing random walks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems-Volume 2*, pages 3077–3085. Curran Associates Inc., 2012.

[27] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[28] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pages 40–48, 2016.

[29] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Hrayr Harutyunyan, Nazanin Alipourfard, Kristina Lerman, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolution architectures via sparsified neighborhood mixing. *arXiv preprint arXiv:1905.00067*, 2019.

[30] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. Graph wavelet neural network. *arXiv preprint arXiv:1904.07785*, 2019.

[31] Fernando Gama, Alejandro Ribeiro, and Joan Bruna. Stability of graph scattering transforms. In *Advances in Neural Information Processing Systems*, pages 8036–8046, 2019.

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

Supplement

A Proofs of Lemma 1 and 2

Proof of Lemma 1. Note that a cyclic graph on $2n$ nodes is a bipartite graph with constant node degree $\beta = 2$. Therefore, the proof of Lemma 1 can be seen as special case of Lemma 2, which is proved below. □

Proof of Lemma 2. We first notice that if $\boldsymbol{g}_{\frac{1}{2}} \star \boldsymbol{x}$ is constant, then $\boldsymbol{g}_\theta \star \boldsymbol{x} = 2\theta(\boldsymbol{g}_{\frac{1}{2}} \star \boldsymbol{x})$ is constant for any θ. Furthermore, for the considered class of graphs, $\boldsymbol{D} = \beta \boldsymbol{I}_n$ with $\beta > 0$, implying that $\boldsymbol{D}^{-1} = \frac{1}{\beta}\boldsymbol{I}_n$ and $\boldsymbol{D}^{-1/2} = \frac{1}{\sqrt{\beta}}\boldsymbol{I}_n$. Therefore, as a direct result of Eq. 1 in the main paper, it holds that

$$\boldsymbol{g}_{\frac{1}{2}} \star \boldsymbol{x} = \left(\frac{1}{2}\boldsymbol{I}_n + \frac{1}{2\beta}\boldsymbol{W}\right)\boldsymbol{x} = \boldsymbol{P}\boldsymbol{x}. \tag{7}$$

Similarly, it is easily verified that any $k \in \mathbb{N}$ applications of the convolution with $\boldsymbol{g}_\theta$ (for any $\theta \in \mathbb{R}$) can be written as $2^k\theta^k\boldsymbol{P}^k\boldsymbol{x}$. Furthermore, since $\boldsymbol{P}$ is column-stochastic and (here) symmetric (thus also row-stochastic), we have $\boldsymbol{P}\boldsymbol{c} = \boldsymbol{c}$ for any constant signal $\boldsymbol{c} = c\boldsymbol{1}_{2n}$. Thus, it is sufficient to show that $\boldsymbol{P}\boldsymbol{x}$ is a constant signal to verify the first claim of the lemma.

We consider the set of nodes V. For any node $v \in V$, according to Eq. 7, we can write

$$(\boldsymbol{P}\boldsymbol{x})[v] = \overbrace{\boldsymbol{P}[v,v]}^{=\frac{1}{2}}\boldsymbol{x}[v] + \sum_{u\in\mathcal{N}(v)} \overbrace{\boldsymbol{P}[v,u]}^{=\frac{1}{2\beta}}\boldsymbol{x}[u] + \sum_{w\in V^v} \overbrace{\boldsymbol{P}[v,w]}^{=0}\boldsymbol{x}[w],$$

where we denote by $\mathcal{N}(v)$ the neighborhood of the node v and set $V^v \coloneqq V \setminus (\{v\} \cup \mathcal{N}(v))$. This implies that

$$(\boldsymbol{P}\boldsymbol{x})[v] = \frac{\boldsymbol{x}[v]}{2} + \sum_{u\in\mathcal{N}(v)} \frac{\boldsymbol{x}[u]}{2\beta}. \tag{8}$$

We now consider a 2-coloring signal $\boldsymbol{x} \in \mathbb{R}^n$. W.l.o.g., let $\boldsymbol{x}[v] = a$, which implies $\boldsymbol{x}[u] = b$ for all $u \in \mathcal{N}(v)$. Now, since $|\mathcal{N}(v)| = \beta$, it holds

$$(\boldsymbol{P}\boldsymbol{x})[v] = \frac{a+b}{2},$$

thus verifying the first claim of the lemma as the choice of v was arbitrary. Finally, it is now straightforward to verify the second claim as well, since the operation $\boldsymbol{\Psi}_0\boldsymbol{x} = (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{x} = \boldsymbol{x} - \frac{a+b}{2}\boldsymbol{1}_n$ retains a 2-coloring signal (the original colors are shifted by a constant: $-\frac{a+b}{2}$). □

B Geometric information encoded by graph wavelets (supp. info. for Sec. 7)

Let $\mathcal{G}_{cyc}^{\geq\ell} \coloneqq \{G_{cyc}^k : k \geq \ell\}$, $3 \leq \ell \in \mathbb{N}$, be the class of unweighted cyclic graphs G_{cyc}^k with lengths $k \geq \ell$. Furthermore, let $\mathcal{G}_\ell^\star$ be the class of graphs constructed by taking $n \geq 2$ cyclic graphs $G_1, \ldots, G_n$, arranging them in the order of the indexes, and connecting subsequent cycles with bottleneck edges as described in the following.

1. We take $G_1, G_n \in \mathcal{G}_{cyc}^{\geq 4}$ and $G_2, \ldots, G_{n-1} \in \mathcal{G}_{cyc}^{\geq\ell}$.
2. The (sub)graph G_i, $1 \leq i \leq n-1$, is connected by exactly one bottleneck edge to G_{i+1}.
3. No edge is connecting (sub)graphs G_i and G_j if $|i-j| \geq 2$.
4. For each G_i, $2 \leq i \leq n-1$, there are exactly two nodes in G_i with bottleneck edges coming out of them, and these nodes are the farthest from each other in the cycle[3] (in shortest-path distance).

[3]For cycles of odd length, the choice of the node to connect is ambiguous (as there are two qualifying nodes), but the claim holds for either choice

This construction essentially generalizes the graph demonstrated in Fig. 2 of the main paper (see Sec. 7). The following lemma shows that on such graphs, the filter responses of $\boldsymbol{g}_\theta$ for a constant signal will encode some geometric information, but will not distinguish between the cycles in the graph. Note that this result can also be generalized further to a chain that is closed by connecting G_n and G_1 with a bottleneck edge if we further assume that $G_1, G_n \in \mathcal{G}_{cyc}^{\geq 7}$.

Lemma 3. *Let $G = (V, E)$ a graph of the class $\mathcal{G}_7^\star$. We consider a constant signal $\boldsymbol{c} = c\mathbf{1}_{|V|}$, for some $c \in \mathbb{R}$. Then, for all nodes $v \in V$ and for any $\theta \in \mathbb{R}$, the filter response $(\boldsymbol{g}_\theta \star \boldsymbol{c})[v]$ shares its value with at least one node of each other cyclic substructure.*

Proof. First, note that V contains only the following two kinds of nodes. We refer to a node of degree 3 (those contained in a bottleneck edge) as a *hub*, while using the term *pass* for all other nodes (those of degree 2). Furthermore, due to the minimal cycle length, and the shortest-path distance requirement between hub nodes in the same cycle, only three types of neighborhoods can be encountered. Indeed, it is easy to see that each hub node has one hub neighbor and two pass neighbors, while pass nodes can either have two pass neighbors or one pass and one hub.

Next, we notice that for any θ and any c, the filter response $(\boldsymbol{g}_\theta \star \boldsymbol{c})[v] = c\theta(\boldsymbol{g}_1 \star \mathbf{1}_{|V|})[v]$ is fully determined by the neighborhood type of $v \in V$. Therefore, computing these boils down to a simple proof by cases:

1. Let v be a hub. Then, the corresponding response is $(\boldsymbol{g}_\theta \star \boldsymbol{c})[v] = c\,\theta\,(1 + \frac{1}{3} + \frac{1}{\sqrt{6}} + \frac{1}{\sqrt{6}}) \approx 2.150 \cdot c\,\theta$.
2. Let v be a pass with two pass neighbors. Then, the corresponding response is $(\boldsymbol{g}_\theta \star \boldsymbol{c})[v] = c\,\theta\,(1 + \frac{1}{2} + \frac{1}{2}) = 2 \cdot c\,\theta$.
3. Let v be a pass with one hub neighbor and one pass neighbor. Then, the corresponding response is $(\boldsymbol{g}_\theta \star \boldsymbol{c})[v] = c\,\theta\,(1 + \frac{1}{\sqrt{6}} + \frac{1}{2}) \approx 1.908 \cdot c\,\theta$.

Finally, it is trivial to see that every cyclic substructure from the class $\mathcal{G}_7^\star$ contains at least one hub and two passes connected to that hub. The existence of a pass only connected to other passes follows from the choice of the minimal cycle lengths together with the requirement that two hubs within a cycle have maximal distance from each other. □

Let us now revisit the example given in the main paper (see Fig. 2). We consider a graph consisting of 14 nodes organized in 2 cycles ($v_1 \sim v_2 \cdots v_6 \sim v_1$ and $v_7 \sim v_8 \cdots v_{14} \sim v_7$ here) of different length (i.e., 6 and 8 here), which are connected with one single edge between any two nodes taken from different cycles (v_4 and v_7 here). As this is a specific case of Lemma 3, the filter responses of the GCN filter $\boldsymbol{g}_\theta$ for a constant signal $\boldsymbol{x}$ would indeed not distinguish between cycles as discussed in Sec. 7 of the main paper (with the pattern shown in Fig. 2 for $\theta = 1$). On the other hand, the filter responses of $\boldsymbol{\Psi}_3\boldsymbol{x}$ on a constant signal (e.g., $\boldsymbol{x} = \mathbf{1}_{14} \in \mathbb{R}^{14}$) on this graph can be verified empirically as follows:

	v_1	v_2	v_3	v_4	v_5	v_6
$\boldsymbol{\Psi}_3\boldsymbol{x}$ ($10^{-3}\times$)	33.0	20.3	-10.7	-52.3	-10.7	20.3

	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
$\boldsymbol{\Psi}_3\boldsymbol{x}$ ($10^{-3}\times$)	-53.5	-13.9	11.2	19.8	19.4	19.8	11.2	-13.9

These responses with appropriate color coding give the illustration in Fig. 2 in the main paper. A similar distinction between cycles with band-pass filters can also be empirically verified for other cases covered by Lemma 3. We leave further theoretical studies of this property of graph wavelets to future work.

C Further discussion of related work

As many applied fields such as Bioinformatics and Neuroscience heavily rely on the analysis of graph-structured data, the study of reliable classification methods has received much attention lately. In this work, we focus on the particular class of semi-supervised classification tasks, where GCN models

[4, 8] recently proved to be effective. Their theoretical studies reveal however that graph convolutions can be interpreted as Laplacian smoothing operations, which poses fundamental limitations on the approach. Another branch of GNNs, manifested in [10], introduces self-attention mechanisms to determine adequate node-wise neighborhoods, which in turn alleviate the mentioned shortcomings of GCN approaches. Further, [9] developed a theoretical framework based on graph signal processing, relying on the relation between frequency and feature noise, to show that GNNs perform a low-pass filtering on the feature vectors. In [29], multiple powers of the adjacency matrix were used to learn the higher-order neighborhood information, while [22] used Lanczos algorithm to construct a low-rank approximation of the graph Laplacian that efficiently gathers multiscale information, demonstrated on citation networks and the QM8 quantum chemistry dataset. Finally, [30] studied wavelets on graphs and collected higher-order neighborhood information based on wavelet transformation.

Together with the study of learned networks, recent studies have also introduced the construction of geometric scattering transforms, relying on manually crafted families of graph wavelet transforms [11–13]. Similar to the initial motivation of geometric deep learning to generalize convolutional neural networks, the geometric scattering framework generalizes the construction of Euclidean scattering from [14] to the graph setting. Theoretical studies [e.g., 11, 31, 23] established energy preservation properties and the stability of these generalized scattering transforms to perturbations and deformations of graphs and signals on them. Moreover, the practical application of geometric scattering to whole-graph data analysis was studied in [12], achieving strong classification results on social networks and biochemistry data, which established the effectiveness of this approach.

As discussed in the main paper, this work aims to combine the complementary strengths of GCN models and geometric scattering and to provide a new avenue for incorporating richer notions of regularity in GNNs. Further, our construction integrates trained task-driven components in geometric scattering architectures. Finally, while most previous work on geometric scattering focused on whole-graph settings, we consider node-level processing, which requires new considerations about the construction.

D Technical details

Similar to other neural networks, the presented Scattering GCN poses several architecture choices and hyperparameters that can be tuned and affect its performance. For simplicity, we set the last layer before the output classification to be the residual convolution layer and only consider one or two hybrid layers before it, each consisting of three GCN channels and two scattering channels. We note that this restricted setup simplifies the network tuning process and was sufficient in our experiments to obtain promising results (outperforming other methods, as shown in the main paper), but can naturally be generalized further to deeper or wider architectures in practice. Furthermore, based on preliminary results, *Cora*, *Citeseer* and *Pubmed* were set to use only one hybrid layer as the addition of a second one was not cost-effective (considering the added complexity of a grid search for tuning hyperparameters based on validation results). For *DBLP*, two layers were used due to a significant increase in performance. We note that even with a single hybrid layer our model achieves 73.1% test accuracy (compared to the reported 81.5% for two layers) and still significantly outperforms GAT (66.1%) and the other methods (below 60%).

Validation & testing procedure: All tests were done using train-validation-test splits of the datasets, where validation accuracy is used for tuning hyperparameters and test accuracy is reported in the comparison table. The same splits were used for all methods for a fair comparison. To ensure our evaluation is comparable with previous work, for *Citeseer*, *Cora* and *Pubmed* we used the same settings as in [4], following the standard practice used in other work reporting results on these datasets. For *DBLP*, as far as we know, no common standard is established in the literature. Here, we used a ratio of $5:1:1$ between train, validation, and test.

Hyperparameter tuning: Given the general network architectures and train-validation-test splits, the hyperparameter tuning was performed for each dataset using grid search guided by validation accuracy. The grid covered the tuning of the redsidual convolution via α, the nonlinearity exponent q (inspired by scattering moments), the scattering channel configuration (i.e., scales used in these two channels), and the widths of channels in the network. The results of this tuning process are presented in the following table.

Table 3: Classification accuracies on Cora with $\alpha = 0.01$ with average accuracy 80.4% over all scales.

ACCURACY	$\boldsymbol{\Psi}_1$	$\boldsymbol{\Psi}_2$	$\boldsymbol{\Psi}_3$	$\boldsymbol{\Psi}_2\|\boldsymbol{\Psi}_3\|$	$\boldsymbol{\Psi}_1\|\boldsymbol{\Psi}_2\|$
$\boldsymbol{\Psi}_1$	0.808	0.808	0.805	0.806	0.806
$\boldsymbol{\Psi}_2$	0.809	0.809	0.806	0.806	0.806
$\boldsymbol{\Psi}_3$	0.802	0.804	0.801	0.801	0.800
$\boldsymbol{\Psi}_2\|\boldsymbol{\Psi}_3\|$	0.802	0.804	0.801	0.800	0.799
$\boldsymbol{\Psi}_1\|\boldsymbol{\Psi}_2\|$	0.802	0.804	0.801	0.800	0.800

Table 4: Classification accuracies on Cora with $\alpha = 0.1$ with average accuracy 80.9% over all scales.

ACCURACY	$\boldsymbol{\Psi}_1$	$\boldsymbol{\Psi}_2$	$\boldsymbol{\Psi}_3$	$\boldsymbol{\Psi}_2\|\boldsymbol{\Psi}_3\|$	$\boldsymbol{\Psi}_1\|\boldsymbol{\Psi}_2\|$
$\boldsymbol{\Psi}_1$	0.813	0.813	0.812	0.808	0.809
$\boldsymbol{\Psi}_2$	0.817	0.817	0.810	0.810	0.815
$\boldsymbol{\Psi}_3$	0.810	0.808	0.801	0.800	0.806
$\boldsymbol{\Psi}_2\|\boldsymbol{\Psi}_3\|$	0.812	0.811	0.801	0.800	0.809
$\boldsymbol{\Psi}_1\|\boldsymbol{\Psi}_2\|$	0.813	0.811	0.802	0.802	0.809

			Scat. config.:		Channel widths:				
	α	q	Φ_{J_1}	Φ_{J_2}	A^1	A^2	A^3	Φ_{J_1}	Φ_{J_2}
Citeseer	0.5	4	$\boldsymbol{\Psi}_2$	$\boldsymbol{\Psi}_2\|\boldsymbol{\Psi}_3\|$	10	10	10	9	30
Cora	0.5	4	$\boldsymbol{\Psi}_1$	$\boldsymbol{\Psi}_3$	10	10	10	11	6
Pubmed	1.0	4	$\boldsymbol{\Psi}_1$	$\boldsymbol{\Psi}_2$	10	10	10	13	14
DBLP(one layer)	1.0	4	$\boldsymbol{\Psi}_1$	$\boldsymbol{\Psi}_2$	10	10	10	30	30
DBLP(two layer)	0.1	1	$\boldsymbol{\Psi}_1$	$\boldsymbol{\Psi}_2$	40	20	20	20	20

It should be noted that for *DBLP*, the hybrid-layer parameters are shared between the two used layers in order to simplify the tuning process, which was generally less exhaustive than for the other three datasets, since even with limited tuning our method significantly outperformed all other methods. That being said, we note that the difference in effectiveness of architecture and hyperparameter choices (as well as the increased performance of our approach compared to others) observed in this case could be a result of the significantly different connectivity exhibited by its graph as discussed briefly in Sec. 8 (of the main paper). Regardless, as more exhaustive tuning would not degrade (and likely improve) the results obtained from Scattering GCN, we view our limited tuning done here as sufficient for establishing the advantages provided by our approach over other methods and leave a more intensive study of the *DBLP* dataset to future work.

Hardware & software environment: All comparisons were executed on the same HPC cluster with intel i7-6850K CPU and NVIDIA TITAN X Pascal GPU. Scattering GCN was implemented in Python using the PyTorch [32] framework. Implementations of all other methods were taken directly from the code accompanying their publications.

E Ablation study

The two main components of our Scattering GCN architecture contribute together to achieve significant improvements over pure GCN models. Namely, these are the additional scattering channels (i.e., Φ_{J_1} and Φ_{J_2}) and the additional residual convolution (controlled by the hyperparemeter α). To further explore their contribution and the hyperparameter space for their tuning, Tab. 3-6 show classification results over the Cora dataset for $\alpha = 0.01, 0.1, 0.5, 1.0$ (controlling the residual convolution layer) over multiple scattering channel configurations. For presentation brevity and simplicity, we focus our presented ablation benchmark on this dataset here, but note that similar results are also observed on the other datasets. The rows and columns in each table denote the two scattering transform channels used in the Scattering GCN, together with the three GCN channels (i.e., for $\boldsymbol{A}^k$, $k = 1, 2, 3$).

First, we consider the importance of the residual graph convolution layer. We note that setting $\alpha = 0$ effectively ignores this layer (i.e., by setting its operation to be the identity), while increasing α makes the filtering provided by it to be more dominant until $\alpha = 1$, where it essentially becomes a random walk-based low-pass filter. Therefore, to evaluate the importance of this component of our architecture, it is sufficient to evaluate the impact of α on the classification accuracy. Indeed, our results (see Tab. 3-6) indicate that increasing α to non-negligible nonzero values improves classification performance, which we interpret to be due to the removal of high-frequency noise. However, when α further increases (in particular when $\alpha = 1$ in this case) the smoothing provided by this layer degrades the performance to a level close to the traditional GCN [4]. Therefore, these results suggest that when well tuned (e.g., as done via grid search in this work), the graph residual convolution plays a critical role in improving results, which can also be seen by the results of the hyperparameter tuning shown in the previous section.

Table 5: Classification accuracies on Cora with $\alpha = 0.5$ with average accuracy 82.7% over all scales.

ACCURACY	Ψ_1	Ψ_2	Ψ_3	$\Psi_2\|\Psi_3\|$	$\Psi_1\|\Psi_2\|$
Ψ_1	0.828	0.828	0.836	0.836	0.835
Ψ_2	0.828	0.833	0.830	0.830	0.827
Ψ_3	0.821	0.829	0.826	0.826	0.826
$\Psi_2\|\Psi_3\|$	0.820	0.828	0.826	0.825	0.825
$\Psi_1\|\Psi_2\|$	0.821	0.829	0.824	0.824	0.824

Table 6: Classification accuracies on Cora with $\alpha = 1.0$ with average accuracy 82.3% over all scales.

ACCURACY	Ψ_1	Ψ_2	Ψ_3	$\Psi_2\|\Psi_3\|$	$\Psi_1\|\Psi_2\|$
Ψ_1	0.817	0.818	0.827	0.827	0.824
Ψ_2	0.820	0.819	0.824	0.824	0.823
Ψ_3	0.826	0.823	0.823	0.823	0.821
$\Psi_2\|\Psi_3\|$	0.825	0.823	0.823	0.823	0.821
$\Psi_1\|\Psi_2\|$	0.822	0.823	0.823	0.823	0.821

Next, we consider the scales used in the two scattering channels of our hybrid architecture, which correspond to the rows and columns of Tab. 3-6 here. While our results show that the network is relatively robust to this choice, we can observe that generally utilizing purely second-order coefficients gives slightly worse results than either first-order ones or a mix of first- and second-order coefficients. Nevertheless, most configurations of scattering scales (with appropriate choice of α) give better results than pure GCN, thus indicating that added information is extracted by scattering channels.

As a final experiment in our ablation study, we present an additional evaluation of $\alpha = 0.35$, which was not included in the original grid search mentioned in the previous section. We clarify that therefore, it was only applied here with the optimal channel widths and q selected previously for *Cora* in the tuning process, while still varying the scattering scales as done in the previous tables.

ACCURACY	Ψ_1	Ψ_2	Ψ_3	$\Psi_2\|\Psi_3\|$	$\Psi_1\|\Psi_2\|$
Ψ_1	0.838	0.836	0.835	0.837	0.837
Ψ_2	**0.842**	0.836	0.835	0.838	0.837
Ψ_3	0.835	0.836	0.833	0.833	0.832
$\Psi_2\|\Psi_3\|$	0.835	0.836	0.833	0.833	0.831
$\Psi_1\|\Psi_2\|$	0.835	0.836	0.832	0.833	0.831

This additional evaluation shows that in this case, all scale configurations outperform GAT (83%) and all other reported methods in Tab. 2. As a result, even the average accuracy over all scale configurations (83.5%) in this case shows an improvement over these other methods, thus further establishing the advantage of our approach. It is important to mention that while this improvement is affected by the tuning of residual convolution, it also relies on the addition of scattering channels. Indeed, by itself, the residual convolution layer only applies a low-pass (smoothing) filter and therefore, without scattering channels, would essentially be equivalent to a conventioal GCN.

We remark that the above table for $\alpha = 0.35$ provides a better configuration than the one provided by our grid search. However, to maintain comparable results with equivalent tuning on all datasets, in Sec. 8 we only report the 83.6% result given by our methodical tuning process, rather than the 84.2% here. However, this ablation attempt also indicates that the presented results set a lower bound on the improvement attained by Scattering GCN, while a more exhaustive optimization of the architecture and its hyperparameters can further improve and solidify its advantages. We leave such exhaustive study for future work, which will also consider the incorporation of other advanced components (e.g., attention mechanisms) in the model architecture.