Scattering GCN: Overcoming Oversmoothness in Graph Conv. Networks

Yimeng Min [*1] **Frederik Wenkel** [*2 1] **Guy Wolf** [2 1]

Abstract

Graph convolutional networks (GCNs) are widely used for semi-supervised node classification on graphs today. The graph structure is however only accounted for by considering the similarity of activations between adjacent nodes, in turn degrading the results. In this work, we augment GCN models by incorporating richer notions of regularity by leveraging cascades of band-pass filters, known as geometric scatterings. We introduce a new hybrid architecture for the task and demonstrate its potential on multiple graph datasets, where it outperforms leading GCN models.

1. Introduction

Geometric deep learning approaches typically use graphs to model geometries, either by constructing them from input data (e.g., via similarity kernels) or directly given as quantified interactions between data points (Bronstein et al., 2017). Using such models, recent works have shown that graph neural networks (GNNs) perform well in multiple fields, including biology, chemistry and social networks (Gilmer et al., 2017; Hamilton et al., 2017; Kipf & Welling, 2016). Most GNNs consider each graph together with given node features as a generalization of images or audio signals, aiming to compute whole-graph representations. These can be applied to graph classification, for example, when each graph represents the molecular structure of proteins or enzymes classified by their chemical properties (Fout et al., 2017; De Cao & Kipf, 2018; Knyazev et al., 2018).

On the other hand, methods such as graph convolutional networks (GCNs) presented by Kipf & Welling (2016) consider node-level tasks, and in particular node classification. As explained in Kipf & Welling (2016), such tasks are often considered in the context of semi-supervised learning, as typically only a small portion of nodes on 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 nodes, which essentially implements local smoothing of neuron activations over the graph (Li et al., 2018). Such smoothing operations often cause degradation of results in node processing tasks due to oversmoothing (Li et al., 2018; NT & Maehara, 2019), as nodes become indistinguishable with deeper and increasingly complex network architectures. Graph attention networks (Veličković et al., 2017) 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 (although 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.

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 (Gama et al., 2018; Gao et al., 2019; Zou & Lerman, 2019), which leverage deep cascades of graph wavelets (Hammond et al., 2011; Coifman & Maggioni, 2006) 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 both GCN and geometric scattering architectures to enable the learning of node-level features that encode geometric information beyond smoothed activation signals, thus alleviating oversmoothing concerns.

Preliminaries: We refer the reader to the supplement for notations and graph signal processing preliminaries.

[*]Equal contribution [1]Mila – Quebec AI Institute, Montréal, QC, Canada [2]Department of Mathematics & Statistics, Université de Montréal, Montréal, QC, Canada. Correspondence to: Yimeng Min <yimeng.min@mail.utoronto.ca>, Guy Wolf <guy.wolf@umontreal.ca>.

Presented at the *ICML 2020 Graph Representation Learning and Beyond (GRL+) workshop*.

2. Graph Convolutional Network

Convolutional filters can be parametrized using the terminology introduced in the supplement. As for graph convolutional networks (GCNs) introduced in Kipf & Welling (2016), we choose just one learnable parameter θ (to avoid overfitting). This is done by setting $\hat{\boldsymbol{g}}[i] \coloneqq \theta(2-\lambda_i)$. The resulting convolutional filtering operation is 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 $[0,2]$, which could lead to vanishing or exploding gradients. This issue is addressed by the following renormalization trick (Kipf & Welling, 2016): $\boldsymbol{I}_n + \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2} \rightarrow \tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{W}}\tilde{\boldsymbol{D}}^{-1/2}$, where $\tilde{\boldsymbol{W}} \coloneqq \boldsymbol{I}_n + \boldsymbol{W}$ and $\tilde{\boldsymbol{D}}$ is a diagonal matrix with $\tilde{\boldsymbol{D}}[i,i] \coloneqq \sum_{j=1}^{n} \tilde{\boldsymbol{W}}[i,j]$ 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 $\boldsymbol{h}_j^\ell = \sigma\big(\sum_{i=1}^{N_{\ell-1}} \theta_i^\ell \boldsymbol{A}\boldsymbol{h}_i^{\ell-1}\big)$, where $\theta_i^\ell \boldsymbol{A}\boldsymbol{h}_i^{\ell-1} = \boldsymbol{g}_{\theta_i^\ell} \star \boldsymbol{h}_i^{\ell-1}$ is the convolutional filter operation. 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 of 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 renormalization 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 (Li et al., 2018) 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.

3. Geometric Scattering

The construction of geometric scattering on graphs 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 renormalizing to retain a Markov process. Therefore, considering an initial distribution $\boldsymbol{\mu}_0 \in \mathbb{R}^n$, its positional distribution after t steps is encoded by $\boldsymbol{\mu}_t = \boldsymbol{P}^t\boldsymbol{\mu}_0$.

As discussed in Gao et al. (2019), the propagation of a graph signal $\boldsymbol{x} \in \mathbb{R}^n$ by $\boldsymbol{x}_t = \boldsymbol{P}^t\boldsymbol{x}$ performs a low-pass operation that suppresses 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 scales 2^k, $k \in \mathbb{N}_0$,

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

for $k \geq 1$. 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 matrices in this filter bank constitutes the identity matrix, thus enabling to reconstruct processed signals from their filter responses. See, e.g., Perlmutter et al. (2019) for further discussion of this construction and its properies.

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 separated by element-wise absolute value nonlinearities, i.e.,

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

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,

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

Here, we modify this construction to keep the scattering transform $\boldsymbol{\Phi}_J$ at the node-level by dismissing the aggregation step in Eq. 4. For each tuple J, we define the scattering propagation rule

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

which mirrors the GCN one while replacing the low-pass filter by a geometric scattering operation. We note that in practice, we only use a subset of tuples, selected as part of our network design explained in the following section.

4. Combining GCN and Scattering Models

To combine the benefits of GCNs and geometric scattering adapted to the node level, we propose a hybrid architecture that combines low-pass operations based on the GCN model with band-pass operations from geometric scattering. To define the layer-wise propagation rule, we introduce

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

and

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

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 $\boldsymbol{A}$, i.e.,

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

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. Inspired by the aggregation step in classical geometric scattering, we use $\sigma(.) \coloneqq |\,.\,|^q$ as nonlinearity. However, unlike in Eq. 4, the q^{th} power is applied at the node-level 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 be the original node features. Each subchannel 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 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. The channel width reflects the importance of the captured regularities. Wider channels suggest that their frequency components are more critical and need to be sufficiently learned.

We note that in sparsely labelled graphs, high-frequency noise (e.g., differences between labelled and unlabelled nodes) can unintentionally be captured by scattering features. To mitigate such artifacts, we introduce the *graph residual convolution*, an adjustable lowpass filter inspired by skip connections in residual networks. Governed by the hyperparameter α , it is defined via the matrix $\boldsymbol{A}_{res}(\alpha) = \frac{1}{\alpha+1}(\boldsymbol{I}_n + \alpha\boldsymbol{W}\boldsymbol{D}^{-1})$ and applied after the hybrid layer of GCN and scattering filters. We have $\boldsymbol{A}_{res}(0) = \boldsymbol{I}_n$, while $\alpha \to \infty$ gives $\boldsymbol{R} = \boldsymbol{W}\boldsymbol{D}^{-1}$, which is an interpolation between the completely lazy random walk $\boldsymbol{I}_n$ 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 is 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 $\boldsymbol{H}^{\ell}$ in Eq. 6. If $H^{\ell+1}$ is the final layer, we set $N_{\ell+1}$ equal to the number of classes.

5. Additional Information Introduced by Node-level Scattering Features

Before empirically verifying the viability of the proposed architecture in node classification tasks, we first discuss a property demonstrating the additional information (in particular, carried by node features) provided by scattering channels beyond that provided by traditional GCN channels. The following lemma shows a particular type of such information in the form of two-coloring features provided on bipartite graphs. We note that such features are highly regular and completely determined by the graph structure, but the smoothing in GCN channels would eliminate their information while the addition of certain scattering channels would retain it for further downstream processing.

Lemma. *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., one part assigned constant a and the other b, where $a \neq b \in \mathbb{R}$). 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. This result extends to any finite linear filter cascade (i.e., $\boldsymbol{g}_\theta \star \cdots \star \boldsymbol{g}_\theta \star \boldsymbol{x}$ or $\boldsymbol{\Psi}_0^k\boldsymbol{x}$ with $k \in \mathbb{N}$ filter applications).*

Table 1. Dataset characteristics

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

6. Empirical Results

To evaluate our Scattering GCN approach, we compare it to several established methods for semi-supervised node classification, including the original GCN (Kipf & Welling, 2016) and the GAT network (Veličković et al., 2017), which indirectly addresses oversmoothing by training adaptive node-wise weighting of the smoothing operation via an attention mechanism, as well as the methods from Li et al. (2018, Partially Absorbing), Defferrard et al. (2016, Chebyshev), and Zhu et al. (2003, Label Propagation). Our comparisons are based on four popular graph datasets[1] with varying sizes and connectivity structures summarized in Tab. 1. We order the datasets here by increasing connectivity structure, reflected by node degrees and edges-to-nodes ratios. As discussed in Li et al. (2018), 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 relative improvement achieved by Scattering GCN to correspond to the increasing connectivity order in Tab. 1, which is maintained in Tab. 2 and Fig. 1.

Table 2. Classification accuracy comparison (top two methods marked in bold; best one underlined) on four benchmark datasets.

MODEL	CITESEER	CORA	PUBMED	DBLP
SCATTERING GCN (OURS)	**71.7**	**84.2**	**79.4**	**81.5**
GAT	**72.5**	**83.0**	79.0	**66.1**
PARTIALLY ABSORBING	71.2	81.7	**79.2**	56.9
GCN	70.3	81.5	79.0	59.3
CHEBYSHEV	69.8	78.1	74.4	57.3
LABEL PROPAGATION	58.2	77.3	71.0	53.0
NODE FEATURES (SVM)	61.1	58.0	49.9	48.2

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 in itself significantly outperforms all other methods.

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. 1 presents the classification accuracy (on validation set) for the training size reduced to 20%, 40%, 60%, 80% and 100% of the original training size available for each dataset. These results indicate that generally Scattering GCN 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 the 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 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.

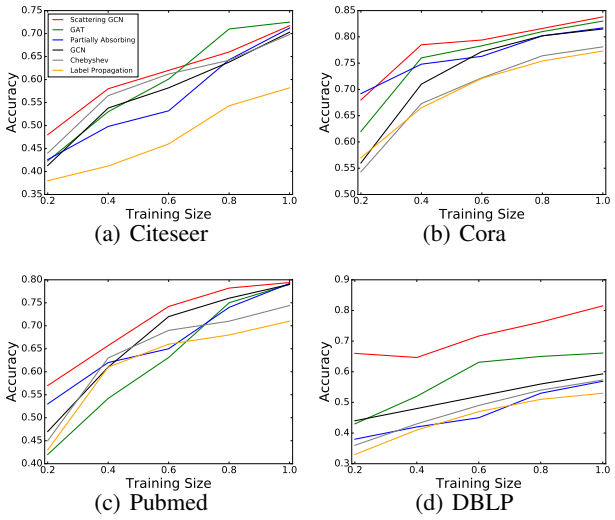

Figure 1. Impact of training set size on classification accuracy.

7. Conclusion

Our study of semi-supervised node classification in graphs presents a new approach to address some of the main concerns and limitations of 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 convolution) in GCN architectures. We expect future work looking to incorporate these elements together in more intricate architectures (e.g., with attention mechanisms) to provide promising new capabilities of pattern recognition and local information extraction in graphs.

[1] See, e.g., Yang et al. (2016) for Citeseer, Cora, and Pubmed, and Pan et al. (2016) for DBLP

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

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34 (4):18–42, 2017.

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

Coifman, R. R. and Maggioni, M. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53 – 94, 2006.

De Cao, N. and Kipf, T. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pp. 3844–3852, 2016.

Fout, A., Byrd, J., Shariat, B., and Ben-Hur, A. Protein interface prediction using graph convolutional networks. In *Advances in neural information processing systems*, pp. 6530–6539, 2017.

Gama, F., Ribeiro, A., and Bruna, J. Diffusion scattering transforms on graphs. *arXiv preprint arXiv:1806.08829*, 2018.

Gao, F., Wolf, G., and Hirn, M. Geometric scattering for graph data analysis. In *International Conference on Machine Learning*, pp. 2122–2131, 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR, 2017.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Hammond, D. K., Vandergheynst, P., and Gribonval, R. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129 – 150, 2011.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International conference on learning representations*, 2016.

Knyazev, B., Lin, X., Amer, M. R., and Taylor, G. W. Spectral multigraph networks for discovering and fusing relationships in molecules. *arXiv preprint arXiv:1811.09595*, 2018.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Liao, R., Zhao, Z., Urtasun, R., and Zemel, R. S. Lanczosnet: Multi-scale deep graph convolutional networks. *arXiv preprint arXiv:1901.01484*, 2019.

NT, H. and Maehara, T. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

Pan, S., Wu, J., Zhu, X., Zhang, C., and Wang, Y. Tri-party deep network representation. *Network*, 11(9):12, 2016.

Perlmutter, M., Gao, F., Wolf, G., and Hirn, M. Understanding graph neural networks with asymmetric geometric scattering transforms. *arXiv preprint arXiv:1911.06253*, 2019.

Shuman, D. I., Ricaud, B., and Vandergheynst, P. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291, 2016.

Susnjara, A., Perraudin, N., Kressner, D., and Vandergheynst, P. Accelerated filtering on graphs using lanczos method. *arXiv preprint arXiv:1509.04537*, 2015.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Yang, Z., Cohen, W. W., and Salakhutdinov, R. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

Zhu, X., Ghahramani, Z., and Lafferty, J. D. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919, 2003.

Zou, D. and Lerman, G. Graph convolutional neural networks via scattering. *Applied and Computational Harmonic Analysis*, 2019.

Supplement

A. 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 $\mathbb{N}_0 \coloneqq \mathbb{N} \cup \{0\}$.

B. Graph Signal Processing Preliminaries

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 $\boldsymbol{D} \coloneqq \operatorname{diag}(d_1, \dots, d_n)$ 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 $\boldsymbol{\Lambda} \coloneqq \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ is a diagonal matrix with the eigenvalues on the main diagonal and $\boldsymbol{Q}$ is an orthogonal matrix containing the corresponding normalized eigenvectors $\boldsymbol{q}_1, \dots, \boldsymbol{q}_n \in \mathbb{R}^n$ as its columns.

A detailed study (see, e.g., (Chung, 1997)) of the eigenvalues reveals that $0 = \lambda_1 \leqslant \lambda_2 \leqslant \dots \leqslant \lambda_n \leqslant 2$. 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 Fourier transform of the convolution of two signals equals the product of the corresponding Fourier transforms. This property generalizes to graphs (Shuman et al., 2016) 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 $\widehat{\boldsymbol{G}} \coloneqq \operatorname{diag}(\hat{\boldsymbol{g}}) = \operatorname{diag}(\hat{\boldsymbol{g}}[1], \dots, \hat{\boldsymbol{g}}[n])$. Hence, convolutional graph filters can be parameterized by using the Fourier coefficients in $\widehat{\boldsymbol{G}}$.

Furthermore, it can be verified (Defferrard et al., 2016) 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 (Kipf & Welling, 2016; Defferrard et al., 2016; Susnjara et al., 2015; Liao et al., 2019) of using filters that have polynomial forms, which we follow here as well.

C. Proof of the Lemma

Proof. 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 $\boldsymbol{c} = c\mathbf{1}_{2n}$. 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

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

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 $\boldsymbol{\Psi}_0\boldsymbol{x} = (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{x} = \boldsymbol{x} - \frac{a+b}{2}\,\mathbf{1}_n$ retains a 2-coloring signal (the original colors are shifted by a constant: $-\frac{a+b}{2}$). □