Multi-Scale Learning on Graphs

Franz Rieger [1]

Abstract

This seminar paper discusses several approaches for multi-scale learning on graphs, their use-cases and trade-offs. Spectral methods like wavelets and scattering transforms can be applied without the need to train and are flexible w.r.t. scale. Spatial methods operating on different scales like graph coarsening, hierarchical clustering or graph neural network (GNN) architectures rely on spatial convolutions in the vertex domain. Some modern methods combine spectral with spatial approaches for better performance.

1. Introduction

Recently, learning on graphs has become an important and active subject of study. It can be applied to a variety of problem statements, such as predicting the spread of a novel disease over a population [19], as well as problems on other real-world graphs like citation networks, brain connectomes, social networks or molecules. In fact, modern GNN approaches as well as classical spectral methods on graphs have made tremendous progress in tackling challenging tasks like semi-supervised node classification, graph classification, link prediction, community detection and the generation of node embeddings. This is reflected in the abundance of survey papers discussing this topic [48; 49; 43; 32; 3].

In this paper we discuss multi-scale learning on graphs. On the one hand, multi-scale can be viewed as the number of nodes in different graphs. Therefore, we review methods which are flexible in the size of the graphs they are dealing with. On the other hand, methods need to operate on different scales when processing a single graph: On the small scale, one can observe a node together with its immediate neighbors. Correspondingly, at bigger scales, further away nodes are relevant. Consequently, at the global scale, the whole graph is observed. Therefore, different kinds of information are processed at different scales. This can be seen intuitively as different levels of abstraction, e.g. in a graph modeling a population different scales refer to individuals, households or cities respectively. This view will prove helpful when discussing hierarchical clustering. An undirected graph can be represented by its N nodes with node features $\mathbf{X} \in \mathbb{R}^{N \times m}$. Its (weighted) edges between the nodes are defined through its symmetric non-negative adjacency matrix $\mathbf{A} \in \mathbb{R}_{+}^{N \times N}$, where $\mathbf{A}_{i,j}$ corresponds to the weight of the edge between the nodes i and j. Real-world graphs are oftentimes sparse, i.e. the number of edges depends linearly on N, which is relevant for computational complexity.

Most of the approaches presented here implicitly rely on the homophily assumption, which states that neighboring nodes are more similar to each other than nodes which are further apart. Oftentimes, these methods on graphs are analogous to existing approaches on euclidean data like images. This especially holds for convolutions [20] whose definition on graphs is not as straightforward as convolutions on image data, where the pixels follow a clear grid structure. As we will see, on graphs we distinguish between spatial and spectral convolutions [5]: **Spatial convolutions** are localized in the vertex domain, where a node in a graph is influenced by its neighbors, much like a pixel's value in an image depends on its surrounding pixels. However, the neighbors cannot be ordered like pixels nor is their count fixed. Therefore, most GNNs perform some form of (repeated) message passing, where the node embeddings of the neighbors are aggregated for each node and then some computation is performed to update the embeddings. Methods working in the spatial domain are discussed in Section 3. **Spectral convolutions** however are localized in the graph's frequency domain as we will see in the next section. They rely on the convolution theorem, which states that a multiplication in the frequency domain corresponds to a convolution in the spatial domain (see Appendix A.2).

2. Spectral Methods

Spectral methods are very useful approaches which oftentimes do not even require any training to work. Similar to Fourier Transformation on Euclidean data[1], Graph Fourier Transformation is applied to transform a graph to the spectral frequency domain. We first define the degree matrix of a graph as $\mathbf{D} = \text{diag}(\sum_{i=1}^{N} \mathbf{A}_{1,i}, \dots, \sum_{i=1}^{N} \mathbf{A}_{N,i})$ and the Laplacian matrix as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Then, the eigenvalues and

[1]Department of Informatics, Technical University of Munich, Munich, Germany. Correspondence to: Franz Rieger <riegerfr@cs.tum.edu>.

[1]https://en.wikipedia.org/wiki/Fourier_transform

eigenvectors ($\mathbf{L}\chi_l = \lambda_l \chi_l$) of the Laplacian are of central importance for spectral methods. The eigenvalues λ_l can be seen as the discrete frequencies of a graph, where the corresponding eigenvector χ_l assigns an amplitude to each node in the graph. A low eigenvalue corresponds to a low frequency, which means that nearby nodes get assigned similar values by the corresponding eigenvector. Accordingly, high eigenvalues correspond to higher frequencies (noise) and the values assigned by the corresponding eigenvector change faster for adjacent nodes. Spectral clustering methods [31] use this property to obtain node embeddings from the corresponding entries in the eigenvectors. Now we can define the graph Fourier transform of a signal on nodes (e.g. the node features $\mathbf{X}$) $f : [N] \to \mathbb{R}^m$ to a signal on eigenvectors $\hat{f}(l) = \sum_{i=1}^{N} \chi_l(i) f(i)$. We can restore the signal to the spatial domain through the inverse which follows as $f(i) = \sum_{l=0}^{N-1} \hat{f}(l) \chi_l(i)$.

Wavelets Spectral graph wavelets, as introduced by Hammond et al. [19] can be represented by a quadratic matrix $\mathbf{\Psi}$. Here, $\mathbf{\Psi}_{i,j}$ denotes how relevant the embedding of node i is for the calculation of the new embedding of node j, which aggregates over all nodes i according to the relevance. This gives a notion for (unlearned) convolution on graphs in the vertex domain, as an alternative to simply just considering the neighbors of a node as GNNs do. With the graph Fourier transform, we can now apply the convolution theorem to define global convolutions in the frequency domain on graphs [5]: A wavelet kernel $g(t\lambda)$ of scale $t > 0$ (which is not to be confused with the corresponding graph wavelet $\mathbf{\Psi}^t$), assigns relevance to different frequencies (eigenvalues). It is utilized to define a spectral graph wavelet transform operator $\widehat{T_g f}(l) = g(\lambda_l)\hat{f}(l)$ by multiplying the signal in the frequency domain. The inverse follows as $(T_g f)(i) = \sum_{l=0}^{N-1} g(\lambda_l)\hat{f}(l)\chi_l(i)$. Note that the explicit wavelet $\mathbf{\Psi}^t_{i,j} = \sum_{l=0}^{N-1} g(t\lambda_l)\chi_l(i)\chi_l(j)$ does not depend on the signal on the graph. With this definition of wavelets, using a hand-selected or learned kernel function $g : \mathbb{R}^+ \to \mathbb{R}$ (which converges to 0 for increasing eigenvalues) and a set of different scales $t_1, \ldots, t_k > 0$ we can now convolve any signal using k different wavelets (of local to global scale) as shown in Figure 1. Different wavelet kernels and scales can also be combined in a filterbank of wavelets. Since the wavelets are usually sparse, this method works faster than graph Fourier Transform with dense eigenvectors.

As discussed by Shuman et al. [38] spectral graph wavelet transform is not the only way to define wavelets. For example, Crovella & Kolaczyk [9] define wavelets based on a wavelet kernel function on the number of hops between a node and the node around which the wavelet is centralized. Other popular wavelets follow different intuitions: Diffusion wavelets [8] analyze the spread of heat through the graph over different timescales, starting at the node they

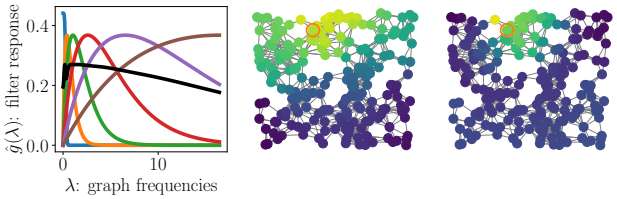

Figure 1. (left) Filterbank of wavelets: sensitivity w.r.t. the graph frequencies λ. (middle and right) Wavelets of different scales localized around the same circled node. Brighter colors correspond to higher relevance for that node. Note how the nodes in the top left corner, a nearby cluster to the selected node, receive high importance for one scale (low frequency), but a lower importance for the other scale (higher frequency). Created with PyGSP.

are localized around. For tree-based wavelets [34] a tree is constructed by grouping together similar datapoints. Average interpolating wavelets [37] are created by interpolating between nodes. Some architectures combine wavelets with learning tasks by making the wavelets learnable [17]. Similar to previous work [5; 21], Defferrard et al. [10] pick up this idea and instead of using a constant wavelet kernel, g is a polynomial on λ with trainable coefficients, which are independent on the graph size. Correspondingly, CayleyNets [26] efficiently use Cayley polynomials for the kernel, which allows learning spectral filters that specialize in the important frequencies or scales.

More modern architectures like Graph Wavelet Neural Network (GWNN) [44] combine spectral wavelets into a GNN for node classification in a sparse, efficient manner, where the convolution is localized in vertex domain. This method is more flexible w.r.t. the neighborhood of a node compared to the previously mentioned node hop wavelets [9]. To compute node embeddings with wavelets, Donnat et al. [13] use heat diffusion wavelets, where the distribution of the nodes' wavelet coefficients is compared to assign similar embeddings to similar distributions. As computing the full eigendecomposition of the Laplacian is computationally expensive, many methods just compute the first few lowest eigenvectors. With the homophily assumption, this approximation is justified because they already convey most of the low noise information, which is needed to reconstruct the rough structure of the graph. In contrast to that, Lanczosnet [30] computes a low rank approximation of the Laplacian using the Lanczos algorithm to efficiently use information from higher frequencies. Which frequencies the model focuses on can again be learned. In contrast to most wavelets, which only depend on the topology of the graph, Rustamov & Guibas [36] also consider the class of signals, which will be defined on the graph, to construct wavelets which can sparsely represent these signals.

Scattering Transforms Graph Scattering Transforms, as defined by Zou & Lerman [51], are an extension of scattering transforms in euclidean space [4]. As a spectral method,

they rely on a filter bank of graph wavelets as convolutional filters, which are applied in multiple layers to form a powerful feature extractor. They transform a signal on a graph by repeatedly applying the wavelet operator, possibly for different scales, to the graph as illustrated in Appendix Figure 3. Then, the embeddings of all the layers can be concatenated for further processing. After reducing the dimensionality of the output and applying a simple MLP, the signal can be used for classification or community detection in graphs. This shows that graph scattering transforms are able to extract important information without training, which is very powerful when little to no training data is available. A similar scattering transform method is described by Gao et al. [16], where the normalized moments of the data are used.

Gama et al. [14] show that diffusion graph scattering transforms are also stable w.r.t. perturbations in the metric domain while still being able to capture high frequency information. Furthermore they are robust to graph manipulations like deformations. Gama et al. [15] also extend graph scattering transforms to multiresolution wavelets, allowing for 'learning' on multiple scales without loosing its robustness property. Alternatively, Chen et al. [7] use haar wavelets for scattering transforms and an unsupervised learning algorithm, which can also perform well on unknown graph topologies, operate on different scales and are again robust to permutations. While the wavelets of Graph Scattering Transforms are usually selected manually and not learned, one possible extension to graph scattering transforms would be also training the wavelet kernels like in CayleyNets.

3. Spatial Methods

In contrast to the spectral methods we saw so far, we now present several approaches for multi-scale learning on graphs, which employ GNNs with spatial convolutions in the vertex domain. These approaches are oftentimes computationally cheaper and perform better than spectral methods. We include methods for coarsening graphs before further processing, hierarchical clustering, as well as a view on different GNN architectures and how they deal with different scales. For a short discussion of methods specialized on specific graphs and sizes refer to Appendix A.3.

Multi-scale GNN architectures Classical GNNs usually aggregate the node embeddings of the direct neighbors. While multiple layers can be stacked, so that also information from multiple hops can reach a node, their performance degrades with increasing depth, which limits the scale of this approach. To address this issue called oversmoothing and to improve performance on larger scales, Multi-hop Hierarchical Graph Neural Networks [46] concatenate the features obtained from multiple message-passing hops. To aggregate the embeddings for all the nodes, attention mechanisms (like proved useful in other domains like Machine Translation [40]) are employed to direct importance to the most relevant nodes in the hops. Similarly, the architecture of Abu-El-Haija et al. [2] also aggregates node embeddings from multiple hops by computing powers of the adjacency matrix in a fast manner. Another approach for using GNNs for learning on a larger scale are residual connections to address the vanishing gradient problem, which occurs when stacking many layers [27].

Instead of using several layers to reach further away information, the approach presented by Klicpera et al. [22] first approximates a personalized PageRank score for every node, which assigns relevance for the other nodes. Then it computes new node embeddings for each node and aggregates them utilizing the PageRank score. Even though this approach allows for infinite depth, the performance does not degrade as the score is localized for every node. The scale of this assignment can be adjusted similar to wavelets by changing the restart probability [33]. Alternatively, Graph diffusion convolution (GDC) [23] can be utilized: It combines spectral approaches with GNNs by transforming the input graph, using diffusion to define the weights of the edges, which are then sparsified. Afterwards any GNN method can be applied on the new graph. Similarly, adaptive graph convolutions [28] learn a residual Laplacian for each graph, which increases local consistency and lets the architecture deal well with different graph sizes. Yet one more notable approach is introduced in Abu-El-Haija et al. [1], where first new edges are discovered by random walks of different lengths, covering different scales. Then, training is performed on the new edges with a combination of different GNNs.

Graph Coarsening Since learning on large graphs is computationally expensive and the problem of oversmoothing appears, some methods first coarsen the graph by removing unimportant nodes while retaining relevant information, such that the graph becomes smaller and learning on multiple scales gets easier. One very recent method using coarsening is GraphZoom [11], which combines several steps to generate node embeddings in an efficient, scalable manner: At first, it performs graph fusion to combine the original adjacency matrix and the node features into a single adjacency matrix. For that, another adjacency matrix is created by using similarity scores of the nodes' embeddings. This new matrix is then combined linearly with the original adjacency matrix to generate the fused graph without node features. To reduce computational complexity, the fused graph is then coarsened by repeatedly merging nodes with high similarity in their spectral embeddings, retaining relevant information (like by Shuman et al. [38]). With the coarsened graph, any node embedding method (spectral or GNN based) can be employed to generate useful node embeddings for the coarsened nodes of the top level. Then, the embeddings are projected back to the original, bottom level nodes by

splitting the coarsened graph's nodes up in the same way they were merged in the coarsening step before. Afterwards, the embeddings are refined using a spectral low pass filter for smoothing.

In comparison to existing methods like MILE [29] which also first coarsen a graph, compute coarse embeddings and then refine them to the original nodes, GraphZoom's refinement is not learned by a GNN model but performed by simple matrix multiplications. This saves training time and also increases performance for some tasks. One possible improvement for GraphZoom could be found in the fusion step: The similarity matrix of the node embeddings is based on a coarsened graph to avoid the quadratic time complexity for the k-nearest-neighbor computation it uses. As a reviewer of the paper noted, existing methods like locality sensitive hashing (LSH) [41] could improve the quality of the fused adjacency matrix and therefore also the overall performance. Since LSH does not take the topology of the graph into account, random walks [50] would be another viable alternative for generating the similarity matrix.

Hierarchical clustering One other way of learning on multiple scales on graphs is hierarchical clustering, which might be viewed as several layers of graph coarsening. The nodes of a graph are assigned to local clusters, which are then repeatedly assigned to (super-)clusters of clusters, which form a hierarchy. In this way, at lower levels close to the original graph, local information can be processed while on higher levels, global information can be passed through the superclusters. As a prominent example, we discuss the Differential Pooling (DiffPool) architecture [47], which has proved especially helpful in graph classification. Internally, DiffPool consists of several layers, which subsequently pool the graph into smaller graphs. Each layer applies an assignment GNN to compute a soft assignment of each node to clusters and an embedding GNN to alter the embeddings. The clusters then build the nodes of the next layer by pooling together the edges and the new embeddings.

One advantage of DiffPool is its flexibility w.r.t. the GNN models. Depending on the use case, simple GNNs (like SAGEConvs [18]) but also advanced methods like GraphZoom for the updated embeddings or a spectral clustering method for the assignments can be applied. With the soft assignment, DiffPool can flexibly learn the cluster assignments in a fully differentiable manner, which allows it to determine the appropriate number of clusters to use. This is an improvement to other methods that depend on a spectral clustering subroutine, which produces a fixed (and therefore possibly sub-optimal) number of clusters [10]. However, the maximal number of clusters in each layer after the initial graph still needs to be defined beforehand, posing difficulties when dealing with a dataset which contains graphs with a high variance in the number of nodes. This could be alleviated by flexibly defining the maximal number of nodes as a ratio of the input nodes of the specific graph. Then, the assignment model is conditioned on multiple random but constant values for each cluster and a softmax function is applied to the assignment scores. Another drawback of DiffPool is its quadratic space complexity in the number of clusters: due to the soft assignment, the pooled graphs are usually fully connected and not sparse anymore.

Therefore, several improvements have been suggested to improve the performance of DiffPool: For reducing the space complexity from quadratic to linear, Cangea et al. [6] adapt the architecture to create sparse pooled layers by dropping nodes instead of pooling nodes together like DiffPool, which leads to fewer, because sparse edges. Which nodes are kept depends only on the amount of information which their embeddings contain. To also incorporate the graph's structure, SAGpool [25] employs an attention mechanism. It uses a GNN model to compute an importance score for every node, which again defines which nodes are kept, in the next layer as illustrated in Appendix Figure 2. One more alternative architecture to DiffPool is EdgePool [12], which uses a different approach to graph pooling by merging the nodes based on edge contractions[2].

4. Discussion and Outlook

We have seen different methods for processing graphs on multiple scales, most of which depend on some form of convolutions on graphs. One category are spectral methods like wavelets and graph scattering transforms where only some rely on training weights. These methods usually require access to the full adjacency matrix and perform convolutions in the frequency domain. The other category are the mostly trainable methods, which rely on convolution in the vertex domain via some form of message passing using GNNs. This includes graph coarsening like GraphZoom, hierarchical clustering with DiffPool or scalable GNN architectures. While the latter perform better in many cases, the spectral methods may not be disregarded [45], when only little or perturbed training data is available, as can be seen in the case of the very flexible graph scattering transforms. With GWNN [44], GDC [23] or the coarsening step in GraphZoom [11], we also saw that many modern approaches combine classical, spectral methods with modern GNN approaches to achieve new state of the art results.

In the future, we can expect more advances in the field of multi-scale learning on graphs through these combinations of deep learning methods with mathematically well founded spectral approaches like the graph wavelet transform. This allows to combine the advantages of spatial and spectral convolutions.

[2] https://en.wikipedia.org/wiki/Edge_contraction

References

[1] Abu-El-Haija, S., Kapoor, A., Perozzi, B., and Lee, J. N-gcn: Multi-scale graph convolution for semi-supervised node classification. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2019. URL http://auai.org/uai2019/proceedings/papers/310.pdf.

[2] Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Steeg, G. V., and Galstyan, A. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing, 2019.

[3] 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.

[4] Bruna, J. and Mallat, S. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[5] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[6] Cangea, C., Veličković, P., Jovanović, N., Kipf, T., and Liò, P. Towards sparse hierarchical graph classifiers. *arXiv preprint arXiv:1811.01287*, 2018.

[7] Chen, X., Cheng, X., and Mallat, S. Unsupervised deep haar scattering on graphs. In *Advances in Neural Information Processing Systems*, pp. 1709–1717, 2014.

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

[9] Crovella, M. and Kolaczyk, E. Graph wavelets for spatial traffic analysis. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, volume 3, pp. 1848–1857. IEEE, 2003.

[10] 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.

[11] Deng, C., Zhao, Z., Wang, Y., Zhang, Z., and Feng, Z. Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1lGO0EKDH.

[12] Diehl, F. Edge contraction pooling for graph neural networks. *arXiv preprint arXiv:1905.10990*, 2019.

[13] Donnat, C., Zitnik, M., Hallac, D., and Leskovec, J. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1320–1329, 2018.

[14] Gama, F., Estrach, J. B., and Ribeiro, A. Diffusion scattering transforms on graphs. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[15] Gama, F., Ribeiro, A., and Bruna, J. Stability of graph scattering transforms. In *Advances in Neural Information Processing Systems*, pp. 8036–8046, 2019.

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

[17] Gavish, M., Nadler, B., and Coifman, R. R. Multiscale wavelets on trees, graphs and high dimensional data: theory and applications to semi supervised learning. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 367–374, 2010.

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

[19] 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.

[20] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[21] Henaff, M., Bruna, J., and LeCun, Y. Deep convolutional networks on graph-structured data, 2015.

[22] Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations (ICLR)*, 2019.

[23] Klicpera, J., Weißenberger, S., and Günnemann, S. Diffusion improves graph learning. In *Advances in*

Neural Information Processing Systems, pp. 13333–13345, 2019.

[24] Klicpera, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs, 2020.

[25] Lee, J., Lee, I., and Kang, J. Self-attention graph pooling. In *International Conference on Machine Learning*, pp. 3734–3743, 2019.

[26] Levie, R., Monti, F., Bresson, X., and Bronstein, M. M. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

[27] Li, G., Müller, M., Thabet, A., and Ghanem, B. Deepgcns: Can gcns go as deep as cnns? In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9266–9275, 2019.

[28] Li, R., Wang, S., Zhu, F., and Huang, J. Adaptive graph convolutional neural networks. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[29] Liang, J., Gurukar, S., and Parthasarathy, S. Mile: A multi-level framework for scalable graph embedding. *arXiv preprint arXiv:1802.09612*, 2018.

[30] Liao, R., Zhao, Z., Urtasun, R., and Zemel, R. S. Lanczosnet: Multi-scale deep graph convo-lutional networks. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[31] Ng, A. Y., Jordan, M. I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pp. 849–856, 2002.

[32] Ortega, A., Frossard, P., Kovačević, J., Moura, J. M., and Vandergheynst, P. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[33] Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[34] Ram, I., Elad, M., and Cohen, I. Generalized tree-based wavelet transform. *IEEE Transactions on Signal Processing*, 59(9):4199–4209, 2011.

[35] Ruiz, L., Gama, F., and Ribeiro, A. Gated graph recurrent neural networks. *arXiv preprint arXiv:2002.01038*, 2020.

[36] Rustamov, R. and Guibas, L. J. Wavelets on graphs via deep learning. In *Advances in neural information processing systems*, pp. 998–1006, 2013.

[37] Rustamov, R. M. Average interpolating wavelets on point clouds and graphs. *arXiv preprint arXiv:1110.2227*, 2011.

[38] Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3): 83–98, May 2013. ISSN 1053-5888. doi: 10.1109/msp.2012.2235192. URL http://dx.doi.org/10.1109/MSP.2012.2235192.

[39] Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pp. 412–422. Springer, 2018.

[40] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[41] Wang, J., Shen, H. T., Song, J., and Ji, J. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.

[42] Wu, F., Jr., A. H. S., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Q. Simplifying graph convolutional networks. In *ICML*, pp. 6861–6871, 2019. URL http://proceedings.mlr.press/v97/wu19e.html.

[43] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[44] Xu, B., Shen, H., Cao, Q., Qiu, Y., and Cheng, X. Graph wavelet neural network. In *7th International Conference on Learning Representations, ICLR 2019*.

[45] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[46] Xue, H., Sun, X.-K., and Sun, W.-X. Multi-hop hierarchical graph neural networks. In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 82–89. IEEE, 2020.

[47] Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In *Advances in neural information processing systems*, pp. 4800–4810, 2018.

[48] Zhang, Z., Cui, P., and Zhu, W. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[49] Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[50] Zhuang, C. and Ma, Q. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 World Wide Web Conference*, pp. 499–508, 2018.

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

A. Appendix

A.1. Additional Figures

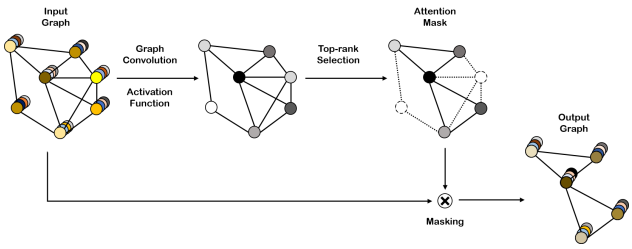

Figure 2. The SAGpool attention mask decides which nodes are kept for the next layer. Figure from [25].

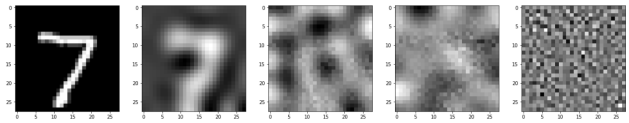

Figure 3. Graph Scattering Transform on a grid graph of pixels. (left to right) input image, then: Graph Scattering Transform layers. Figure from [51].

A.2. Convolution theorem

The convolution theorem[3] states that convolving the kernel g over a signal f is equivalent to first transforming g and f to frequency domain using $\mathcal{F}$, usually using a Fourier transformation, to achieve $\hat{g} = \mathcal{F}(g)$ and $\hat{f} = \mathcal{F}(f)$. Then the convolution in the spatial domain is equivalent to a multiplication in the frequency domain:
$f * g = \mathcal{F}^{-1}(\mathcal{F}(f) \cdot \mathcal{F}(g))$

[3] https://en.wikipedia.org/wiki/Convolution_theorem

A.3. Other methods

RecurrentGNNs [35] can be used for graphs modeling dynamic processes (like the initially mentioned spread of a novel disease) and therefore enable scaling in the time domain.

With scale as the size of the graph, the approaches vary with the number of nodes: For small graphs, different architectures are possible than for large graphs [39; 18]. Learning on small graphs (like molecules) also allows for much more complex GNN architectures which are tuned for that task [24] in contrast to the simple architectures like SAGEConv or the model of Wu et al. [42] employed for larger graphs.