Spectral-designed depthwise separable graph neural networks

Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, Paul Honeine

Spectral-designed Depthwise Separable Graph Neural Networks

Balcilar Muhammet [1] **Renton Guillaume** [1] **Héroux Pierre** [1] **Gaüzère Benoit** [2] **Adam Sébastien** [1] **Honeine Paul** [1]

Abstract

This paper aims at revisiting Convolutional Graph Neural Networks (ConvGNNs) by designing new graph convolutions in spectral domain with a custom frequency profile while applying them in the spatial domain. Within the proposed framework, we propose two ConvGNNs methods: one using a simple single-convolution kernel that operates as a low-pass filter, and one operating multiple convolution kernels called Depthwise Separable Graph Convolution Network (DSGCN). The latter is a generalization of the depthwise separable convolution framework for graph convolutional networks, which allows to decrease the total number of trainable parameters while keeping the capacity of the model unchanged. Our proposals are evaluated on both transductive and inductive graph learning problems, demonstrating that DSGCN outperforms the state-of-the-art methods.

1. Introduction

Convolutional Neural Networks (CNNs) had a significant impact in machine learning for signal and image processing. Most successes have been realized on data defined on grid Euclidean spaces. However, there are many domains where data cannot be encoded into an Euclidean space, but are naturally represented as graphs, such as with molecules and social networks. For this end, Graph Neural Networks (GNNs) have been recently investigated (Gilmer et al., 2017; Bronstein et al., 2017; Wu et al., 2019).

Convolutional GNNs (ConvGNNs) seek to extract features through a weight-sharing strategy, in the same spirit as CNNs. In a nutshell, the graph convolution process corresponds to the multiplication of a convolution kernel with the corresponding node feature vectors, followed by a sum or a mean rule. Two strategies have been proposed to design filter kernels, either in the spectral or in the spatial domains.

[1]LITIS Lab, University of Rouen Normandy, Rouen, France. [2]LITIS Lab, INSA Rouen, Rouen, France. Correspondence to: Balcilar Muhammet <muhammetbalcilar@gmail.com>.

ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+), Vienna, Austria.

Spectral-based convolution relies on the spectral graph theory. Despite the solid mathematical foundations borrowed from the signal processing literature, such approaches suffer from (i) the computational burden of the forward/inverse graph Fourier transform, (ii) being spatially non-localized and (iii) the transferability problem, i.e., filters designed using a given graph cannot be applied on other graphs. To alleviate these issues, several reparameterizations have been introduced, such as with B-spline (Bruna et al., 2013), Chebyshev polynomials (Defferrard et al., 2016) and Cayley polynomials (Levie et al., 2019). However, these parametrizations cannot extract band specific information.

Spatial-based convolutions aggregate nodes neighborhood information, in the same spirit as the conventional Euclidean convolution (e.g. 2D convolution in CNNs). Such convolutions are very attractive due to their low computational complexity, their localized property and their transferability. However, their spectral behavior is not the same for different graphs and there is no guaranty to extract useful information on different frequency bands.

In this paper, we propose to design graph convolution in spectral domain with a custom frequency profile. We assume the number of the convolutions and their frequency profiles are hyperparameter of the model and are to be tuned during validation process. Within the considered framework, we propose two ConvGNN methods: A method with a single convolution kernel that operates as a low-pass filter (denoted LowPassConv), and a method based on multiple convolution kernels called Depthwise Separable Graph Convolution Network (DSGCN). The latter allows to decrease the total number of trainable parameters while keeping the variability capacity of the model at a maximum level. The concept of depthwise separable convolution was recently introduced in computer vision problems to reduce the model's complexity (Chollet, 2017; Sandler et al., 2018), but has not been investigated for ConvGNN so far. The proposed methods LowPassConv and DSGCN are assessed on both transductive and inductive learning problems (Yang et al., 2016). In both settings, we show the relevance of the proposed methods on well-known public benchmark datasets. Especially, the success of the proposed methods on inductive problems provides one of the first experimental evidence of transferability of spectral filter coefficients from one graph to another.

2. A Primal on ConvGNNs

Spectral ConvGNNs are defined using a graph signal processing. For a given graph, let U be the eigenvectors matrix of its graph Laplacian L, and $\boldsymbol{\lambda}$ the vector of its eigenvalues. A graph convolution layer in spectral domain can be written as a sum of filtered signals followed by an activation function σ (e.g. RELU) as in (Bruna et al., 2013), namely

$$H_j^{(l+1)} = \sigma\Big(\sum_{i=1}^{f_l} U\,\mathrm{diag}(F_{i,j,l})U^\top H_i^{(l)}\Big), \qquad (1)$$

for all $j \in \{1,\dots,f_{l+1}\}$, where $H_j^{(l)}$ is the j-th feature vector of the l-th layer, and $F_{i,j,l}$ is the trainable weight vector. This formulation is intractable since it requires computing the graph Fourier transform and its inverse, by matrix multiplication of U and U^T. Another drawback is the filter non-transferability in multi-graph learning problems. To overcome these issues, it is often re-parameterized as

$$F_{i,j,l} = B\left[W_{i,j}^{(l,1)},\dots,W_{i,j}^{(l,S)}\right]^\top, \qquad (2)$$

where $B \in \mathbb{R}^{n\times S}$ is the parameterization matrix, n is number of nodes in the given graph, S is the desired number of convolution kernels and $W^{(l,s)}$ is the trainable matrix with index $s = 1..S$. Each column in B is designed as a function of eigenvalues, i.e., $B_{i,j} = F_j(\boldsymbol{\lambda}_i)$, such as with B-spline (Bruna et al., 2013), Chebyshev polynomials (Defferrard et al., 2016) and Cayley polynomials (Levie et al., 2019).

Recent research (Balcilar et al., 2020) shows that spectral ConvGNN in (1) parametrized by (2) can be written in spatial ConvGNN as follows:

$$H^{(l+1)} = \sigma\Big(\sum_s C^{(s)} H^{(l)} W^{(l,s)}\Big), \qquad (3)$$

with the convolution kernel set to

$$C^{(s)} = U\,\mathrm{diag}(F_s(\boldsymbol{\lambda}))U^\top, \qquad (4)$$

where $diag$ creates a diagonal matrix by vector.

3. Proposed Methods

Instead of designing the spatial convolution kernels $C^{(s)}$ of (3) by functions of structural properties of graph (spatial approach) or some predefined functions (B-spline, Polynomial, Chebyshev, Cayley) of eigenvalues (existing spectral approach), we propose to use S convolution kernels that have custom-designed standard frequency profiles. These designed frequency profiles are a function of eigenvalues, such as $[F_1(\boldsymbol{\lambda}),\dots,F_S(\boldsymbol{\lambda})]$. In this proposal, the number of kernels and their frequency profiles are hyperparameters. Then, we can back-compute the corresponding spatial convolution matrices using (4).

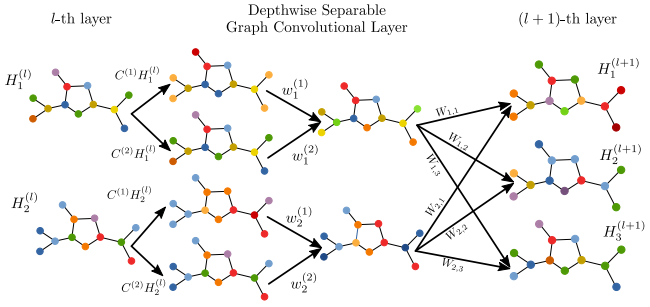

Figure 1. Detailed schematic of Depthwise Separable Graph Convolution Layer. Each node has a 2-length feature vector, indicated as $H_1^{(l)}$ and $H_2^{(l)}$ with values represented by colors. The following layer has a 3-length feature vector, denoted $H_1^{(l+1)}$, $H_2^{(l+1)}$ and $H_3^{(l+1)}$. Here, two convolution kernels are used, denoted by $C^{(1)}$ and $C^{(2)}$. Convoluted signals are multiplied by trainable weight w and are summed to obtain interlayer signals. To obtain the 3 next layer features, a weighted sum is computed using the other trainable parameter W.

3.1. LowPassConv

To evaluate the performance of convolutions designed in the spectral domain independently from the architecture design, a single hidden layer is used for all models, as in (Kipf & Welling, 2017) for GCN. This choice, even sub-optimal, enables a deep understanding of the convolution kernels.

For this purpose, we propose a single convolution kernel that operates as a low-pass filter (denoted LowPassConv), with a profile defined by

$$F_1(\boldsymbol{\lambda}) = (\mathbf{1} - \boldsymbol{\lambda}/\lambda_{\max})^\eta, \qquad (5)$$

where η is a tunable parameter influencing the cut-off frequency. The choice of using a low-pass filter comes from the fact that state-of-the-art GNNs are sort of low-pass filters and perform well on the well-known datasets. Therefore, this model can be seen as similar to those from (Defferrard et al., 2016; Kipf & Welling, 2017) but using a different convolution kernel.

3.2. DSGCN

In order to figure out arbitrary relations of input-output pairs, multiple convolution kernels have to be efficiently designed. Indeed, to obtain problem-agnostic graph convolutions, the sum of all designed convolutions' frequency profiles has to cover most of the possible spectrum and each kernel's frequency profile must focus on some certain ranges of frequencies. However, increasing the number S of convolution kernels increases the number of trainable parameters linearly. Hence, the total number of multi-support ConvGNN is given by $S\sum_{i=0}^{L} f_i f_{i+1}$ where L is the number of layers and f_i is the feature length of the i-th layer.

To overcome this issue, we propose to use Depthwise Separable Graph Convolution Network (DSGCN). The Depthwise Separable Convolution framework was recently proposed in computer vision problems to reduce the model size and its complexity (Chollet, 2017; Sandler et al., 2018). To the best of our knowledge, depthwise separable graph convolution has never been proposed in the literature.

Instead of filtering all input features for each output feature, DSGCN filters each input feature once. Then, filtered signals are merged into the desired number of output features through 1×1 convolutions with different contribution coefficients. Illustration of the proposed depthwise separable graph convolution process is presented in Figure 1.

Mathematically, forward calculation of each layer of DSGCN is defined by:

$$H^{(l+1)} = \sigma\left(\Big(\sum_{s=1}^{S} w^{(s,l)} \odot (C^{(s)}H^{(l)})\Big)W^{(l)}\right), \quad (6)$$

where $\odot$ denotes the element-wise multiplication operator. Note that there is only one trainable matrix W in each layer. Other trainable variables $w^{(s,l)} \in \mathbb{R}^{1\times f_l}$ encode feature contributions for each convolution kernel and layer. The number of trainable parameters for this case is $\sum_{i=0}^{L} Sf_i + f_i f_{i+1}$. Previously, adding a new kernel increases the number of parameters by $\sum_{i=0}^{L} f_i f_{i+1}$. Using separable convolutions, this number is only increased by $\sum_{i=0}^{L} f_i$. This modification is particularly interesting when the number of features is high. On the other hand, the variability of the model also decreases. If the data has a smaller number of features, using this approach might not be optimal.

4. Experimental Evaluation

In this section, we describe the experiments carried out to evaluate the proposed approach on both transductive and inductive problems.

4.1. Transductive Learning Problem

Experiments on transductive problems were led on three well-known datasets: Cora, Citeseer and PubMed (summary in Table A1 in Appendix). These datasets are well-known paper citation graphs. Each node corresponds to a paper. If one paper cites another one, there is an unlabeled and undirected edge between the corresponding nodes. Binary features on the nodes indicate the presence of specific keywords in the corresponding paper. The task is to attribute a class to each node (i.e., paper) of the graph using for training the graph itself and a very limited number of labeled nodes. We use predefined train, validation and test sets as defined in (Yang et al., 2016) and follow the test procedure of (Kipf & Welling, 2017; Veličković et al., 2018) for fair comparisons.

Obtained results are given in Table 1 in terms of accuracy. We compare the performance of the proposed LowPassConv and DSGCN to state-of-the-art methods. We first can see that our low-pass convolution kernel (LowPassConv) obtains comparative performance with existing methods. It is worth noting that the good results obtained by low-pass approaches show that these three classification tasks are mainly low-pass specific problems. Differences in accuracies may be significantly bigger for band-pass or high-pass based problems. Moreover, SGCN outperforms state-of-the-art methods thanks to the flexibility provided by the different filters.

4.2. Inductive Learning Problem

Inductive Learning problems are common in chemoinformatics and bioinformatics. In an inductive setting, a given instance is represented by a single graph. Thus, models are trained and tested on different graph sets. These experiments are led on 3 datasets (see Table A4 in Appendix for a summary): a multi-graph node classification dataset called Protein-to-Protein Interaction (PPI) (Zitnik & Leskovec, 2017) and two graph classification datasets called PROTEINS and ENZYMES (Kersting et al., 2016). The protocols used for the evaluations are those defined in (Veličković et al., 2018) for PPI and (Ying et al., 2018; Cangea et al., 2018; Ting Chen, 2019; Xu et al., 2019) for PROTEINS and ENZYMES datasets.

The PPI dataset is a multi-label node classification problem on multi-graphs. Each node has to be classified either True or False for 121 different criteria. All the nodes are described by a 50-length continuous feature vector. The PPI dataset includes 24 graphs, with a train/validation/test standard splitting.

The PROTEINS and ENZYMES datasets are graph classification datasets. There are 2 classes in PROTEINS and 6 classes in ENZYMES. In PROTEINS dataset, there are three different types of nodes and one continuous feature. But we do not use this continuous feature on nodes. In ENZYMES dataset, there are 18 continuous node features and three different kinds of node types. In the literature, some methods use all provided continuous node features while others use only node label. This is why ENZYMES results are given using either all features (denoted by ENZYMES-allfeat) or only node labels (denoted by ENZYMES-label). Since there is no standard train, validation and test sets split for PROTEINS and ENZYMES, the results are given using a 10-fold cross-validation (CV) strategy under a fixed predefined epoch number. The CV only uses training and validation set. Specifically, after obtaining 10 validation curves corresponding to 10 folds, we first take average of validation curves across the 10 folds and then select the single epoch that achieved the maximum averaged validation

Table 1. Comparison of methods. All results are accuracy percentage, while PPI results are micro-F1 metric percentage.

	Cora	Citeseer	Pubmed	PPI	PROTEINS	ENZYMES-label	ENZYMES-allfeat
MLP ($C^{(1)} = I$)	55.1	46.5	71.4	46.2 ± 0.56	74.03 ± 0.92	27.83 ± 2.51	76.11 ± 0.87
GraphSAGE (Hamilton et al., 2017)	-	-	-	76.8	-	-	-
GCN (Kipf & Welling, 2017)	81.9 ± 0.5	70.7 ± 0.4	78.9 ± 0.3	59.2 ± 0.52	75.12 ± 0.82	51.33 ± 1.23	75.16 ± 0.65
GAT (Veličković et al., 2018)	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.7	97.3 ± 0.20	-	-	-
ChebNet (Defferrard et al., 2016)	81.2	69.8	74.4	-	75.50 ± 0.40	58.00 ± 1.40	-
CayleyNet (Levie et al., 2019)	81.9 ± 0.7	-	-	-	-	-	
DPGCNN (Monti et al., 2018)	83.3 ± 0.5	72.6 ± 0.8	-	-	-	-	
MoNet (Monti et al., 2017)	81.7 ± 0.5	-	78.8 ± 0.3	-	-	-	
GaAN (Zhang et al., 2018)	-	-	-	98.7 ± 0.20	-	-	-
Hierarchical (Cangea et al., 2018)	-	-	-	-	75.46	64.17	-
Diffpool (Ying et al., 2018)	-	-	-	-	76.30	62.50	66.66
Multigraph (Knyazev et al., 2018)	-	-	-	-	76.50 ± 0.40	61.70 ± 1.30	68.00 ± 0.83
GIN (Xu et al., 2019)	-	-	-	-	76.20 ± 0.86	-	-
GFN (Ting Chen, 2019)	-	-	-	-	76.56 ± 0.30	60.23 ± 0.92	70.17 ± 0.86
LowPassConv (this paper)	0.827 ± 0.006	0.717 ± 0.005	0.794 ± 0.005	58.6 ± 0.47	74.81 ± 0.78	52.21 ± 1.06	74.84 ± 0.71
DSGCN (this paper)	$\mathbf{84.2 \pm 0.5}$	$\mathbf{73.3 \pm 0.8}$	$\mathbf{81.9 \pm 0.3}$	$\mathbf{99.09 \pm 0.03}$	$\mathbf{77.28 \pm 0.38}$	$\mathbf{65.13 \pm 0.65}$	$\mathbf{78.39 \pm 0.63}$

accuracy. This procedure is repeated 20 times with random seeds and random division of dataset. Mean accuracy and standard deviation are reported. This is the same protocol than (Ying et al., 2018; Ting Chen, 2019; Xu et al., 2019; Cangea et al., 2018).

Table 1 compares the results obtained by the models described above and state-of-the-art methods. A comparison with the same models but without graph information, a Multi-Layer Perceptron (MLP) that corresponds to $C^{(1)} = I$ is also provided to discuss if structural data include information or not. To the best of our knowledge, such analysis is not provided in the literature for transductive setting problems. Finally, results obtained by the same architecture with GCN kernel is also provided.

As one can see in Table 1, the proposed method DSGCN obtains competitive results on inductive datasets. For PPI, DSGCN clearly outperforms state-of-the-art methods with the same protocol, reaching a micro-F1 percentage of 99.09 and an accuracy of 99.45%. For this dataset, MLP accuracy is low since the percentage of micro-F1 is 46.2 (random classifier's micro-F1 being 39.6%). This means that the problem includes significant structural information. Using the GCN kernel, the accuracy increases to 0.592, but again not comparable with state-of-the-art accuracy.

For the PROTEINS dataset, MLP (namely $C^{(1)} = I$) reaches an accuracy that is quite comparable with state-of-the-art GNN methods, with 74.03% validation accuracy, while the proposed DSGCN reaches 77.28%, which is the best performance among GNNs. This means that PROTEINS problem includes very few structural information to be exploited by GNNs.

The ENZYMES dataset results allows to understand the importance of continuous features and their processing through different convolutions. As one can see in Table 1, there are important differences of performance between the results on ENZYMES-label and ENZYMES-allfeat. When node labels are used alone, without features, MLP accuracy is very poor and nearly acts as a random classifier. When using all features, MLP outperforms GCN and even some state-of-the-art methods. A first explanation is that methods are generally optimized for just node label but not for continuous features. Another one is that the continuous features already include information related to the graph structure since they are experimentally measured. Hence, their values are characteristic of the node when it is included in the given graph. Since GCN is just a low-pass filter, it removes some important information on higher frequency and decreases the accuracy. Thanks to the multiple convolutions proposed in this paper, DSGCN clearly outperforms other methods on the ENZYMES dataset.

5. Conclusion

In this paper, we investigated a framework for designing new graph convolutions in spectral domain with a custom frequency profile, while applying them in the spatial domain. Two ConvGNNs were proposed, LowPassConv as a single-convolution low-pass kernel, and multiple-convolution DSGCN that decreases the number of trainable parameters while keeping the model's capacity unchanged. Extensive experiments carried out on well-known datasets showed the relevance of the proposed methods, with DSGCN outperforming the state-of-the-art methods.

Acknowledgments

This work was partially supported by the ANR grant APi (ANR-18-CE23-0014) and the PAUSE Program.

References

Balcilar, M., Renton, G., Héroux, P., Gaüzère, B., Adam, S., and Honeine, P. Bridging the gap between spectral and spatial domains in graph neural networks. *arXiv preprint arXiv:2003.11702*, 2020.

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, July 2017.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203, 2013.

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

Chollet, F. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

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.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyal, O., and Dahl, G. E. Neural message passing from quantum chemistry. In *Proceedings of the International Conference on Machine Learning*, 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.

Kersting, K., Kriege, N. M., Morris, C., Mutzel, P., and Neumann, M. Benchmark data sets for graph kernels, 2016. http://graphkernels.cs.tu-dortmund.de.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

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.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

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, Jan 2019. ISSN 1941-0476. doi: 10.1109/TSP.2018.2879624.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5115–5124, 2017.

Monti, F., Shchur, O., Bojchevski, A., Litany, O., Günnemann, S., and Bronstein, M. M. Dual-primal graph convolutional networks, 2018.

Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 4510–4520. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00474.

Ting Chen, Song Bian, Y. S. Dissecting graph neural networks on graph classification. *CoRR*, abs/1905.04579, 2019.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596, 2019.

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

Yang, Z., Cohen, W. W., and Salakhutdinov, R. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning, ICML'16*, 2016.

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.

Zhang, J., Shi, X., Xie, J., Ma, H., King, I., and Yeung, D.-Y. Gaan: Gated attention networks for learning on large and

spatiotemporal graphs. In *Conference on Uncertainty in Artificial Intelligence, UAI*, 2018.

Zitnik, M. and Leskovec, J. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33 (14):i190–i198, 2017.

Supplementary Material

Spectral Designed Depthwise Separable Graph Neural Networks

Muhammet Balcilar, et al.

A. Application Details

Transductive Learning Problem

Dataset: Experiments on transductive problems were led on the three datasets summarized in Table A1 where each dataset consists of one single graph.

Models: To evaluate the performance of convolutions designed in the spectral domain independently from the architecture design, a single hidden layer is used for all models, as in (Kipf & Welling, 2017) for GCN. This choice, even sub-optimal, enables a deep understanding of the convolution kernels. For these evaluations, a set of convolution kernels is experimented by

- A low-pass filter defined by $F_1(\boldsymbol{\lambda}) = (\mathbf{1} - \boldsymbol{\lambda}/\lambda_{\max})^{\eta}$ where η impacts the cut-off frequency
- A high-pass filter defined by $F_2(\boldsymbol{\lambda}) = \boldsymbol{\lambda}/\lambda_{\max}$
- Three band-pass filters defined by:
  - $F_3(\boldsymbol{\lambda}) = \exp(-\gamma(0.25\lambda_{\max} - \boldsymbol{\lambda})^2)$
  - $F_4(\boldsymbol{\lambda}) = \exp(-\gamma(0.5\lambda_{\max} - \boldsymbol{\lambda})^2)$
  - $F_5(\boldsymbol{\lambda}) = \exp(-\gamma(0.75\lambda_{\max} - \boldsymbol{\lambda})^2)$
- An all-pass filter defined by $F_6(\boldsymbol{\lambda}) = \mathbf{1}$

We firstly consider a model composed of only F_1. This choice comes from the fact that state-of-the-art GNNs are sort of low-pass filters and perform well on the datasets of Table A1. Hence, it is interesting to evaluate our framework with F_1. For the experiments, the value of η are tuned for each dataset, using the validation loss value and accuracy, yielding $\eta = 5$ for Cora and Citeseer, and $\eta = 3$ for PubMed. Details concerning this tuning can be found in Table A2. Since there is only one convolution kernel, depthwise separable convolutions are not necessary for this model. Therefore, this model can be seen as similar to those from (Defferrard et al., 2016; Kipf & Welling, 2017) but using a different convolution kernel. This approach is denoted as LowPassConv in Table 1.

Beyond this low-pass model, we also evaluate different combinations of the $F_i(\boldsymbol{\lambda})$ through the depthwise separable schema defined in Section 3. For experiments involving $\{F_3(\boldsymbol{\lambda}), F_4(\boldsymbol{\lambda}), F_5(\boldsymbol{\lambda})\}$, the bandwidth parameter γ was tuned using train and validation sets. Table A3 details the best models found on the validation set. As an example, for Cora dataset, 4 kernels are used by a DSGCN with 160 neurons: $F_1(\boldsymbol{\lambda})$, $F_3(\boldsymbol{\lambda})$, $F_4(\boldsymbol{\lambda})$, $F_5(\boldsymbol{\lambda})$. As an illustration, Figure 2 provides the standard frequency profiles of this designed convolution on Cora dataset. The models of Table A3 are denoted as DSGCN in the following.

The training hyperparameters were tuned over a grid search using a cross-validation procedure. Hyperparameter values can be found in Table A6.

Table A1. Summary of the transductive datasets used in our experiments.

	Cora	Citeseer	PubMed
# Nodes	2708	3327	19717
# Edges	5429	4732	44338
# Features	1433	3703	500
# Classes	7	6	3
# Training Nodes	140	120	60
# Validation Nodes	500	500	500
# Test Nodes	1000	1000	1000

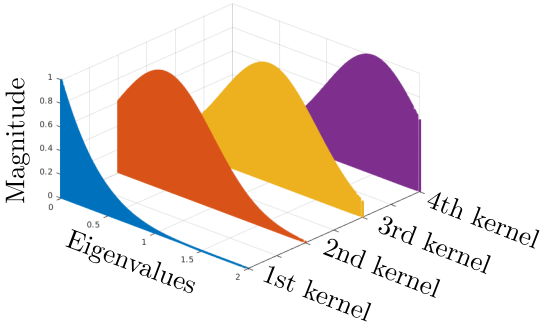

Figure 2. Designed convolution's frequency profiles for Cora dataset.

Inductive Learning Problem

Dataset: Experiments on inductive problems were led on the three datasets summarized in Table A4 where each dataset consists of different number of graphs.

Table A2. Minimum validation set loss value and maximum validation set accuracy over different low-pass filters.

Convolution	Cora		Citeseer		PubMed	
$F_1(\boldsymbol{\lambda})$	Loss	Acc	Loss	Acc	Loss	Acc
$(\mathbf{1}-\boldsymbol{\lambda}/\lambda_{\max})^1$	1.116	80.4	1.12	73.0	0.654	77.1
$(\mathbf{1}-\boldsymbol{\lambda}/\lambda_{\max})^3$	0.745	81.8	1.02	72.6	**0.572**	**81.1**
$(\mathbf{1}-\boldsymbol{\lambda}/\lambda_{\max})^5$	**0.705**	**81.8**	**1.02**	**73.8**	0.592	80.5
$(\mathbf{1}-\boldsymbol{\lambda}/\lambda_{\max})^{10}$	0.752	81.2	1.01	72.2	-	-
$(\mathbf{1}-\boldsymbol{\lambda}/\lambda_{\max})^{20}$	0.792	80.8	1.01	71.2	-	-

Table A3. Used kernels frequency profiles and architecture of models for each transductive dataset. DSG refers to Depthwise Separable Graph convolution layer, G to Graph convolution layer, D to Dense layer

Dataset	Architecture
Cora	$F_1(\boldsymbol{\lambda}) = (\mathbf{1}-\boldsymbol{\lambda}/\lambda_{\max})^5$ $F_3(\boldsymbol{\lambda}) = \exp(-0.25(0.25\lambda_{\max}-\boldsymbol{\lambda})^2)$ $F_4(\boldsymbol{\lambda}) = \exp(-0.25(0.5\lambda_{\max}-\boldsymbol{\lambda})^2)$ $F_5(\boldsymbol{\lambda}) = \exp(-0.25(0.75\lambda_{\max}-\boldsymbol{\lambda})^2)$ DSG160-DSG7
Citeseer	$F_1(\boldsymbol{\lambda}) = (\mathbf{1}-\boldsymbol{\lambda}/\lambda_{\max})^5, F_6(\boldsymbol{\lambda}) = \mathbf{1}$ DSG160-DSG6
Pubmed	$F_1(\boldsymbol{\lambda}) = (\mathbf{1}-\boldsymbol{\lambda}/\lambda_{\max})^3, F_2(\boldsymbol{\lambda}) = \boldsymbol{\lambda}/\lambda_{\max}$ DSG16-DSG3

Models: For PPI, 7 depthwise graph convolution layers compose the model. Each layer has 800 neurons, except the output layer which has 121 neurons, each one classifying the node either True or False. All layers use a ReLU activation except the output layer, which is linear. No dropout or regularization of the binary cross-entropy loss function is used. All graph convolutions use three spectral designed convolutions: a low-pass convolution given by $F_1(\boldsymbol{\lambda}) = \exp(-\boldsymbol{\lambda}/10)$, a high-pass one given by $F_2(\boldsymbol{\lambda}) = \boldsymbol{\lambda}/\lambda_{\max}$ and an all-pass filter given by $F_3(\boldsymbol{\lambda}) = \mathbf{1}$.

For graph classification problems (PROTEINS and ENZYMES), depthwise graph convolution layers are not needed since these datasets have a reduced number of features. Thus, it is tractable to use all multi-support graph convolution layers instead of the depthwise schema. In these cases, our models firstly consist of a series of graph convolution layers. Then, a global pooling (i.e., graph readout) is applied in order to aggregate extracted features at graph level. For this pooling, we use a concatenation of mean and max global pooling operator, as used in (Cangea et al., 2018). Finally, a dense layer (except for ENZYMES-label) is applied, before the output layer as in (Xu et al., 2019).

All details about the architecture and designed convolutions can be found in Table A5. The hyperparameters used in best models can be found on Table A6.

Table A4. Summary of inductive learning datasets used in this paper.

	PPI	PROTEINS	ENZYMES
Type	Node Class.	Graph Class.	Graph Class.
# Graph	24	1113	600
# Avg.Nodes	2360.8	39.06	32.63
# Avg.Edges	33584.4	72.82	62.14
# Features	50	3 label	3 label + 18 cont.
# Classes	2 (121 criterias)	2	6
# Training	20 graphs	9-fold	9-fold
# Validation	2 graphs	1-fold	1-fold
# Test	2 graphs	None	None

Table A5. Kernels frequency profiles and model architecture for each inductive dataset. meanmax refers to global mean and max pooling layer.
Same legend as Table A3.

Dataset	Architecture
PPI	$F_1(\boldsymbol{\lambda}) = \exp(-\boldsymbol{\lambda}/10)$ $F_2(\boldsymbol{\lambda}) = \boldsymbol{\lambda}/\lambda_{\max}, F_3(\boldsymbol{\lambda}) = \mathbf{1}$ DSG800-DSG800-DSG800-DSG800- DSG800-DSG800-DSG121
PROTEINS	$F_1(\boldsymbol{\lambda}) = \mathbf{1}-\boldsymbol{\lambda}/\lambda_{\max}, F_2(\boldsymbol{\lambda}) = \boldsymbol{\lambda}/\lambda_{\max}$ G200-G200-meanmax-D100-D2
ENZYMES-label	$F_1(\boldsymbol{\lambda}) = \mathbf{1}, F_2(\boldsymbol{\lambda}) = \boldsymbol{\lambda}_s - \mathbf{1}$ $F_3(\boldsymbol{\lambda}) = 2\boldsymbol{\lambda}_s^2 - 4\boldsymbol{\lambda}_s + \mathbf{1}, \boldsymbol{\lambda}_s = 2\boldsymbol{\lambda}/\lambda_{\max}$ G200-G200-G200-G200-meanmax-D6
ENZYMES-allfeat	$F_1(\boldsymbol{\lambda}) = \mathbf{1}, F_2(\boldsymbol{\lambda}) = \exp(-\boldsymbol{\lambda}^2)$ $F_3(\boldsymbol{\lambda}) = \exp(-(\boldsymbol{\lambda}-0.5\lambda_{\max})^2)$ $F_4(\boldsymbol{\lambda}) = \exp(-(\boldsymbol{\lambda}-\lambda_{\max})^2)$ G200-G200-meanmax-D100-D6

HYPERPARAMETERS

In depthwise separable graph convolution layer, the initialization of the trainable parameters $w^{(s,l)}$ affects the performance. If designed convolutions are supposed to have equal effect on the model, these parameters can be initialized randomly. But, if one is supposed to have more effect on the model, the important convolution kernel's correspondence weights can be initialized by 1, the rest of them initialized by 0. In our model, we assumed the first kernel is always the most important kernel. Thus, we initialized the first kernel's depthwise separable weights as $w^{(1,l)} = 1$, and the rest of the kernel's depthwise separable weights $w^{(s,l)} = 0$ when $s > 1$. In this way, the model starts training as there is only kernel, which is supposed to be the most important one.

The used hyperparameters in our experiments are presented in Table A6. We applied `softmax` to the output of the models and calculate cross entropy loss function for all problems expect PPI dataset. Since PPI is two class classification problem and we coded output by one neuron, we applied `tansig` to the output of the PPI model and used binary cross entropy as loss function. In our models we

Table A6. Used hyperparameters.

Hyperparameters	Cora	Citeseer	PubMed	PPI	PROTEINS	ENZYMES-label	ENZYMES-allfeat
Hidden Activations	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU
Output Activation	Linear	Linear	ReLU	Linear	Linear	Linear	Linear
Hidden Biases	False	False	False	True	False	False	False
Output Bias	True	True	False	True	True	True	True
Input Dropout	0.75	0.75	0.25	0	0	0.1	0.1
Kernel Dropout	0.75	0.75	0	0	0	0.1	0.1
Weight Decay	3e-4	3e-4	5e-4	0	0	1e-4	1e-4
Weight Decay on DSG	3e-3	3e-3	5e-3	0	-	-	-
Learning Coeff	0.01	0.01	0.01	0.01	0.0005	0.001	0.001
Batch Size	1	1	1	1	333	180	180
Epoch	400	100	250	500	100	500	500

did not consider any regularization on the bias parameter, but we applied the L2-loss to the trainable weights. In the depthwise separable layer, there are two different kinds of weights where additional one is depthwise weights. That is why in Table A6, there is two different weight decays. We always used `ReLU` activation on the hidden layers and the Linear for output layers. The table also provides if bias values are used in the hidden and output layers. In our model, we used two different types of dropout: the dropout applied on the inputs of the layer as usually used in the literature, and the dropout applied on the convolution kernel, which was first used in (Veličković et al., 2018) according to the best of our knowledge. Since Cora, Citeseer and PubMed datasets consist of one single graph, batch size is 1 for these problems. For the PPI dataset of only 24 graphs, we still prefer to update the model for each training graph. But for PROTEINS and ENZYMES datasets, we update the model 3 times in each epoch. Since in PROTEINS there are 1113 graphs and in each fold there are 1000 graphs in the train set, we used a 333 batch size. As the same in ENZYMES there is 540 graphs in each train fold, we used a 180 batch size to update the model 3 times in a single epoch. We used the Adam optimization and a fixed learning-coefficient in all models. The used learning coefficient and the maximum epoch number can be found in the Table A6.