Unleashing the Potential of Dilated Convolution and Edge Features for GCN

Nathan Corecco, Giorgio Piatti, David Gu,
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—Graph Convolutional Networks (GCN) are a popular approach for analyzing graph-structured data, but they can suffer from the issue of underreachability, where nodes that are far apart are unable to exchange information. We propose a new message passing framework for GCNs called dilated message passing, which overcomes the issue of underreachability by applying dilated convolutions to increase the receptive field of each node, allowing for information to exchange between nodes that were previously too far apart. Additionally, we develop a new stochastic method for creating edge features in the dilated convolution process. Our work provides new insights into the use of dilated convolution and advanced edge feature handling for analyzing graph data, and tries to address the problems of underreachability and oversquashing and significantly improves the performance of GCNs on a variety of tasks.**

I. Introduction

Graphs are a powerful modeling tool, capable of representing complex systems such as chemical compounds or social networks. Graph Convolutional Networks (GCN) have demonstrated effectiveness in tasks such as node classification and graph classification. However, they are also susceptible to various challenges commonly known as oversquashing and underreachability. Specifically, underreachability refers to the difficulty in exchanging information between nodes that are far apart, while oversquashing pertains to the loss of important information due to excessive amount of information averaged into a single message. To address these issues, we propose a new method for GCN called dilated message passing, inspired by the field of computer vision where dilated convolutions have had a significant impact on the performance of Convolutional Neural Networks (CNN) [1]. In our approach, the message passing phase of GCN is split into two parts. First, nodes exchange messages with their neighbors as in the traditional GCN setting [2]. Afterwards, we apply dilated convolutions to increase the receptive field of each node, allowing for information exchange between nodes that were previously too far apart. Additionally, we develop a new stochastic method for creating edge features in the dilated convolution process. By combining dilated convolutions with our edge feature generation method, we aim to alleviate both the problem of underreachability and the issue of oversquashing [3], where important information is lost due to excessive filtering. Our proposed dilated message passing method has the potential to significantly improve the performance of GCNs on a wide variety of tasks.

II. Related Work

There has been a growing body of research on the use of graph convolutional neural networks (GCNNs) for analyzing graph-structured data. Early work in this area focused on adapting traditional convolutional neural network (CNN) architectures to irregular graphs, by defining a local neighborhood for each node and using this neighborhood to perform convolution [4], [2]. More recent approaches have sought to improve upon these methods by introducing new concepts from graph theory, such as spectral graph convolution [5] and graph attention [6] mechanisms. Li et al. [7] studied depth limitation of GCNs and showed that deep GCNs can cause over-smoothing, i.e. features at vertices in a connected component tend to converge to the same value, influencing the performance as the number of layers increases. Aron et al. [3] systematically analyzed how GNNs are susceptible to bottlenecks when aggregating messages across a long path, leading to over-squashing of exponentially increasing information into fixed-size vectors. This bottleneck hinders popular message passing GNNs architectures from fitting long-range signals. They propose a simple solution to break the bottleneck and improved the state-of-the-art result on real world datasets without any tuning or additional weights, thus concluding that further research may be done in solving over-squashing. Recently transformers, which are a type of attention-based model originally developed for natural language processing [8], [9], have been applied to GCN [10], allowing for the incorporation of both spatial and relational information in a single model.

Dilated convolution [1] has also been explored as a method for expanding the receptive field of CNNs on regular grids, with the goal of improving performance on image tasks. The work of Li et. al [11] proposed a method based on dilated convolution neural network to perform point cloud segmentation on point cloud represented as euclidean graphs. However, the application of dilated convolution to GCNNs has received relatively little attention.

In this paper, we aim to fill this gap by introducing the use of dilated convolution in GCNNs for general graphs, and demonstrate its effectiveness for improving the accuracy and efficiency of graph-based learning tasks. We also present a novel approach for constructing edge features in a stochastic fashion, which allows to capture more nuanced relationships between nodes. Our work offers a new perspective on the use of dilated convolution and advanced edge feature

construction for analyzing graph data.

III. Models and methods

In this section, we present our proposed model for graph analysis. The model consists of two steps. In the first step, we perform a standard Message passing phase: Messages are exchanged between nodes to compute an embedding for each node that contains information about its local neighborhood. In the second step, we perform Dilated message passing, inspired by dilated convolution in CNNs [1]: Nodes exchange messages with nodes further away in order to increase the receptive field and create more expressive feature vectors.

First, we introduce some basic notation.

Graph Notation: We represent a graph G as a tuple (V, E), where V is the set of nodes and E is the set of edges which connect the nodes. For each node $v \in V$, we represent its feature vector as $x_v \in \mathbb{R}^d$. Moreover we define $N_l(v)$ as the set of nodes in G that are at distance l away from v. Similarly, for each edge $\{v, u\} \in E$, we represent its feature vector as $e_{v,u} \in R^{d'}$. We also denote by h_G the feature vector for the entire graph. In our setting, h_G is the concatenation of all the node feature vectors, i.e. $h_G = [x_{v_1}, \ldots, x_{v_n}]$.

Message Passing Phase: The first phase consists of k_1 message passing layers. A message passing layer, which generalizes convolution to irregular domains, is expressed as a neighborhood aggregation scheme. For each node $v \in V$ with feature vector $x_v^{(l-1)}$ in layer $l-1$, we update its embedding as follows:

$$x_v^{(l)} = \gamma_{\theta_l}^{(l)} \left(x_v^{(l-1)}, \psi_{u \in N_1(v)} \phi_{\theta_l}^{(l)} \left(x_v^{(l-1)}, x_u^{(l-1)}, e_{v,u} \right) \right), \tag{1}$$

where ψ is a differentiable, permutation invariant function that aggregates the features of the neighboring vectors, and γ and ϕ are differentiable functions such as multi-layer perceptrons. γ combines the previous feature vectors $x_v^{(l-1)}$ with the result of the aggregation to compute the new feature vector, and ϕ is used to combine the feature vectors of the neighbors with the edge features.

Dilated message passing phase: For a node $v \in V$ after k_1 steps of normal message passing its feature vector $x_v^{(k_1)}$ contains the information of all nodes in $N_{\leq k_1}(v) := \bigcup_{i \in [k_1]} N_i(v)$, all nodes at distance at most k_1 from v. In the dilated convolution phase, a node v, instead of exchanging a message with its close neighborhood $N_1(v)$ exchanges a message with all nodes in N_{2k_1}, in order to speed up its "exploration process". After the message exchange its feature vector $x_v^{(k_1+1)}$ contains information about all nodes in $N_{\leq 3k_1}$. In the next step, it aggregates messages from its $N_{6k_1}(v)$ neighborhood and the resulting feature vector will contain the information about all nodes in $N_{\leq 9k_1}$. Thus the receptive field of each node v will grow exponentially. Formally, for a graph G, after k_1 layers of normal message passing layers, the l^{th} dilated graph convolution layer updates the nodes' feature vectors as follows:

$$a_v^l = \psi_{u \in N_{2k_1 \cdot 3^{l-1}}(v)} \phi_{\theta_l}^{(l)} \left(x_v^{(l-1)}, x_u^{(l-1)}, \hat{e}_{v,u} \right) \tag{2}$$

$$x_v^{(l)} = \gamma_{\theta_l}^{(l)} \left(x_v^{(l-1)}, a_v^{(l)} \right). \tag{3}$$

This formula resembles Equation (1): $a_v^{(l)}$ is the result of the aggregation, which then gets combined via γ to obtain the new feature vector. There are two differences though: The first one is that this time the message gets exchanged with the nodes in $N_{2k_1 \cdot 3^{l-1}}(v)$ instead of the nodes in $N_1(v)$. The second difference concerns the edge feature vector: Since we are not considering the classical neighborhood $N_1(v)$ there is no actual edge between u and v. To this end we construct the edge feature $\hat{e}_{u,v}$ by randomly sampling a path of length $2k_1 \cdot 3^{l-1}$ between u and v (which exists since $u \in N_{2k_1 \cdot 3^{l-1}}(v)$) and the feature vector $\hat{e}_{u,v}$ is obtained by aggregating the feature vectors of all edges on the path. More formally, let $p = (e_1, \ldots, e_m)$ a vertex disjoint path sampled at random [1] between v and u, then the aggregated edge vector $\hat{e}_{v,u}$ used in the layer is constructed using GEM pooling as follows:

$$(\hat{e}_{v,u})_i = \left[\left(\frac{1}{m} \sum_{j \in [m]} (e_j)_i^q \right)^{\frac{1}{q}} \right]. \tag{4}$$

GEM pooling was presented by Radenović et al. in "Fine-tuning CNN image retrieval with no human annotation" [12]. It uses a learnable parameter $q > 0$, for $q = 1$ the pooling equals an average pooling, while for $p \to \infty$ it converges to a max-pooling. In this way we are able to create an embedding for connections not in E. This stochastic approach to construct edge embeddings not only helps in improving generalization, but also allows to construct a representative embedding without having to compute all possible paths of length m, which in some cases can grow exponentially in m. When edge features are not present, we can speed up the learning process by precomputing the necessary neighborhoods. If edge features are present, random paths need to be computed in every epoch as described.

Strategies for Enhancing Performance: Since in our dilated message passing setting there is the possibility that a node does not completely collect all the information about its dilated neighborhood, we use two strategies. First, we slightly augment the dilated message passing phase in the l^{th} layer by allowing every node v to exchange messages with nodes in the set

$$N_{2k_1 \cdot 3^{l-1}}(v) \cup \left(N_{(k_1 \cdot 3^{l-1}, 2k_1 \cdot 3^{l-1})} \cap L_{\text{BFS}}(v) \right), \tag{5}$$

[1] We implicitly sample the path uniformly at random by sampling an ordering of the neighbors and subsequently calling BFS.

where $L_{\text{BFS}}(v)$ represents the set of leaves in the BFS tree rooted at v, and $N_{(k_1 \cdot 3^{l-1}, 2k_1 \cdot 3^{l-1})}$ is the set of nodes at a distance greater than $k_1 \cdot 3^{l-1}$ and less than $2k_1 \cdot 3^{l-1}$. In addition, similar to "Attention is All You Need" [8], we add a length encoding to the path embedding in order to account for varying distances of the aggregated message along the path.

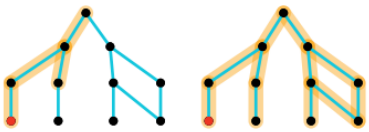

Figure 1: The illustration depicts the receptive field of the red leaf bottom left, for both message passing frameworks. Left, the receptive field of a traditional MP framework after three message exchanges, while right it is illustrated the receptive field of the proposed dilated MP framework, which includes two rounds of standard message passing followed by one round of dilated message passing.

Residual connections: The use of residual connections was first proposed by He et al. [13] and has greatly improved the ability of deeper architectures to train effectively. In our model we included two types of residual connections to improve the flow of gradients and prevent vanishing. These connections are placed between consecutive dilated message passing layers and between the feature vectors after normal and dilated message passing steps, allowing the network to effectively utilize both local and global information when constructing final node embeddings.

IV. Datasets

In our experiments, we evaluate the performance of our approach on several graph classification datasets, ENZYMES, PROTEINS, NCI1, DD and BZR-MD and COX2-MD, some of which contain edge features. All aforementioned datasets can be downloaded from the TU Dataset collection [14]. To further validate our approach on edges, we use ogbg-molhiv [15]. We refer to the Appendix A for dataset statistics. To test the effect on over-squashing, we use the synthetic benchmark Tree-Neighboursmatch, designed by Alon et al. [3]. It consists of a binary tree of variable depth K, rooted at v. The problem is a classification task, to predict a label for the root, where the right answer is the label of a leaf node that has the same one-hot feature vector. We refer to the original paper [3] for more information on the dataset and the specific details.

V. Experiments

In this study, we evaluate the performance of our proposed message passing framework for GCN through extensive experiments on various datasets. We compare our framework to classical graph neural network models, using a diverse range of datasets and implementing our framework for each of them. We create 10 data splits for each dataset, using 80% for training and dividing the remaining 20% between validation and test. Furthermore, for each dataset and for every layer type , we adhere to the guidelines outlined by Errica et al. [16] to fine-tune the hyperparameters and ensure a fair comparison metric among different message passing techniques. All the hyperparameters are listed in Appendix C. In the latter portion of this section, we investigate the performance of our model on the synthetic dataset proposed by Alon et al. [3], which is designed to evaluate the efficiency of different models when increasing the radius of the graph and, as a result, how much they are affected by oversquashing and underreaching.

A. Comparison of Proposed Model

In this section, we evaluate our message passing framework against the traditional approach by utilizing six datasets and four types of aggregation functions. The accuracy metric is used for comparison and the hyperparameter search was in accordance with the guidelines of [16].

GIN Layer: Our initial experiment uses a GIN-layer [17] to aggregate messages. We evaluate the standard message passing model as well as our proposed framework with and without leaf node consideration across six different datasets.

Dataset	MP	Dilated	Dilated with leaves
BZR-MD	**67.0** $\pm$ 8.49	64.33 $\pm$ 8.95	64.33 $\pm$ 8.95
COX2-MD	63.67 $\pm$ 4.82	**66.33** $\pm$ 4.82	**66.33** $\pm$ 4.82
DD	72.99 $\pm$ 2.48	74.78 $\pm$ 3.47	**75.9** $\pm$ 5.68
ENZYMES	64.0 $\pm$ 7.04	**65.67** $\pm$ 4.96	64.67 $\pm$ 4.46
NCI1	80.51 $\pm$ 1.87	80.42 $\pm$ 1.5	**80.68** $\pm$ 1.2
PROTEINS	73.6 $\pm$ 5.34	**75.5** $\pm$ 5.16	73.51 $\pm$ 4.21

Table I: Test Accuracy on layer GIN

For the GIN-layer, we conduct an experiment using a larger dataset. The performance metric used for this dataset was ROC-AUC instead of accuracy. It's worth noting that the data splits for this dataset were provided by the problem statement [15] and not created by our team. For this dataset the distribution of the test data slightly differs from the validation and training distributions, making generalization more challenging.

Dataset	MP [15]	Dilated	Dilated with leaves
Validation	82.32 $\pm$ 00.90	**84.9** $\pm$ 1.04	84.77 $\pm$ 0.55
Test	75.58 $\pm$ 01.40	76.56 $\pm$ 1.43	**77.49** $\pm$ 0.32

Table II: ROC-AUC on layer GIN for ogbg-molhiv

GAT Layer: The second experiment that we conduct uses a GAT-layer [6]. As previously we have the results across the six datasets, for our two models and the classical one. [2]

Dataset	MP	Dilated	Dilated with leaves
BZR-MD	**69.67** $\pm$ 8.36	67.67 $\pm$ 5.78	67.67 $\pm$ 5.78
COX2-MD	**73.33** $\pm$ 8.3	69.0 $\pm$ 6.51	69.0 $\pm$ 6.51
DD	72.82 $\pm$ 4.06	**76.07** $\pm$ 5.17	75.38 $\pm$ 4.34
ENZYMES	71.0 $\pm$ 5.78	**72.0** $\pm$ 3.78	70.17 $\pm$ 6.68
NCI1	79.59 $\pm$ 1.58	79.51 $\pm$ 1.67	**80.66** $\pm$ 1.91
PROTEINS	73.34 $\pm$ 4.11	**74.96** $\pm$ 4.75	74.06 $\pm$ 4.64

Table III: Test Accuracy on layer GAT

[2]Due to high requirement of GPU RAM, we couldn't complete a full grid search, for the model dilated with leaves with GAT on the DD dataset.

GraphSAGE Layer: In our third experiment, we use a GraphSAGE Layer to aggregate messages[18].

Dataset	MP	Dilated	Dilated with leaves
BZR-MD	75.67 ± 6.51	**76.34** ± 9.6	**76.34** ± 9.6
COX2-MD	69.0 ± 8.83	**72.33** ± 9.2	67.67 ± 6.15
DD	75.64 ± 4.56	**76.07** ± 3.99	75.47 ± 2.29
ENZYMES	61.67 ± 6.19	**68.33** ± 4.15	66.0 ± 4.84
NCI1	80.95 ± 1.85	81.31 ± 1.84	**81.58** ± 1.84
PROTEINS	72.97 ± 3.8	**75.14** ± 4.82	73.51 ± 7.29

Table IV: Test Accuracy on layer GraphSage

GCN Layer: The last layer that we use for comparison in our experiment is the classical GCN layer first proposed by Kipf et al. [4].

Dataset	MP	Dilated	Dilated with leaves
BZR-MD	68.67 ± 4.52	**71.0** ± 8.44	**71.0** ± 8.44
COX2-MD	**64.0** ± 9.16	63.67 ± 6.23	63.67 ± 6.23
DD	73.5 ± 4.6	**75.64** ± 4.82	75.21 ± 4.52
ENZYMES	57.17 ± 6.37	**60.5** ± 4.41	**60.5** ± 4.41
NCI1	73.92 ± 2.09	**75.06** ± 1.99	73.89 ± 2.43
PROTEINS	**75.76** ± 5.09	74.23 ± 5.88	74.68 ± 3.29

Table V: Test Accuracy on layer GCN

B. Results from Synthetic Dataset Evaluation

In our final experiment, we evaluate the effectiveness of our proposed dilated message passing framework in addressing the problem of oversquashing using a synthetic dataset introduced by Alon et al. [3]. This dataset allows for the generation of graphs with varying radii, with larger radii posing a more difficult task due to increased distance and number of classes. The goal of this dataset is not generalization but overfitting. We train our model using GIN-layer aggregation on problems from radius 2 to 7, and compare it to the results obtained by traditional MP. For the purpose of comparison, we use a selection of well-established models including GIN [17], GAT [6], GCN [4], and GGNN [19], which were previously trained and tested on this dataset by Alon et al. [3].

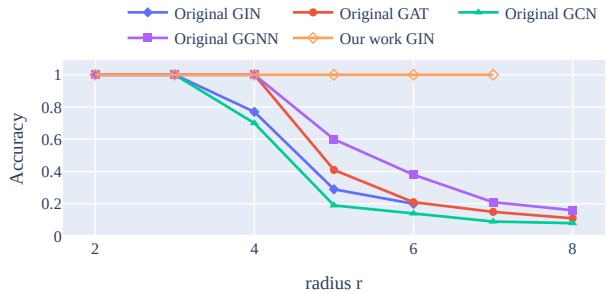

Figure 2: Comparison of model performance across varying problem radius, with accuracy plotted on the y-axis.

As discussed in the original paper, larger values on the hidden dimensionality for standard message passing lead to the same trend. For reference we present the best hyperparameters for each radius in appendix B.

VI. Discussion

As observed from the results presented in the previous section, several key insights can be gleaned. The comparison between the traditional architecture and our proposed architecture utilizing the GIN-layer reveals that, out of the six datasets, the dilated message passing approach outperforms the traditional approach in five of them. Using the GAT-layer, the proposed approach performs better than the standard approach in four out of six tasks. In the experiments conducted utilizing the GraphSAGE-layer, our proposed message passing framework outperforms the traditional approach across all six datasets. Finally, when utilizing the GCN-layer, our method demonstrates superior performance in four out of six experiments. Hence overall, our approach outperforms traditional message passing in nineteen out of twenty-four settings. On average, it improves the accuracy of the traditional message passing by 1.19%, indicating that the implementation of dilated message passing significantly improves performance. Moreover, a significant enhancement can be observed on the obgb-molhiv dataset, where the implementation of the random path sampling method appears to aid in better generalization and results in improved scores on the test set. Specifically, both variations of the dilated message passing outperform the traditional approach on both the validation and test splits, with an average improvement of 2.6% accuracy on the validation set and 2% accuracy on the test set.

The results of the final experiment, utilizing the synthetic dataset, illustrates the most drastic improvement achieved by the proposed dilated message passing framework. Specifically, the accuracy of the GCN and GIN models begins to decline rapidly from a radius at four. The GAT and GGNN models exhibit slightly better performance, but still experience a significant drop in accuracy for radii larger than four. In contrast, our model is able to maintain an accuracy of 1 even for problem radii up to seven [3], demonstrating the effectiveness of our approach in addressing the issue of oversquashing, particularly for small radius problems, outperforming all the standard models.

VII. Summary

Our proposed message passing framework that uses dilated convolution and random path sampling improves the performance of GCNs. We conducted a variety of experiments on various datasets and found that our approach outperforms traditional methods. Moreover, we also tested our model on a synthetic dataset designed to evaluate its ability to mitigate oversquashing and observed significant improvement over standard message passing frameworks.

[3]The missing measurement for a radius of 8 is due to lack of resources (GPU and time)

REFERENCES

[1] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[2] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.

[3] U. Alon and E. Yahav, "On the bottleneck of graph neural networks and its practical implications," *arXiv preprint arXiv:2006.05205*, 2020.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[5] H. Zhu and P. Koniusz, "Simple spectral graph convolution," in *International Conference on Learning Representations*, 2020.

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

[7] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[10] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," *Advances in neural information processing systems*, vol. 32, 2019.

[11] G. Li, M. Muller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9267–9276.

[12] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning cnn image retrieval with no human annotation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 7, pp. 1655–1668, 2018.

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

[14] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," *arXiv preprint arXiv:2007.08663*, 2020.

[15] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *Advances in neural information processing systems*, vol. 33, pp. 22 118–22 133, 2020.

[16] F. Errica, M. Podda, D. Bacciu, and A. Micheli, "A fair comparison of graph neural networks for graph classification," *arXiv preprint arXiv:1912.09893*, 2019.

[17] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[18] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[19] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[20] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *Journal of molecular biology*, vol. 330, no. 4, pp. 771–783, 2003.

[21] N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," *Knowledge and Information Systems*, vol. 14, no. 3, pp. 347–375, 2008.

[22] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[23] I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, and D. Schomburg, "Brenda, the enzyme database: updates and major new developments," *Nucleic acids research*, vol. 32, no. suppl_1, pp. D431–D433, 2004.

[24] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl_1, pp. i47–i56, 2005.

[25] J. J. Sutherland, L. A. O'brien, and D. F. Weaver, "Spline-fitting with a genetic algorithm: A method for developing classification structure- activity relationships," *Journal of chemical information and computer sciences*, vol. 43, no. 6, pp. 1906–1915, 2003.

[26] S. Zhang and L. Xie, "Improving attention mechanism in graph neural networks via cardinality preservation," in *IJCAI: Proceedings of the Conference*, vol. 2020. NIH Public Access, 2020, p. 1395.

VIII. Appendix

A. Dataset statistics

Dataset	# Graphs	# Classes	# Avg. Nodes	# Avg. Edges	# Node features	# Edge features
DD [20], [21], [22]	1178	2	284.32	715.66	89	-
ENZYMES [23], [24]	600	6	32.63	64.14	21	-
NCI1 [22]	4110	2	29.87	32.30	37	-
PROTEINS [20], [24]	1113	2	39.06	72.82	4	-

Table VI: Dataset overview with node-based features

Dataset	# Graphs	# Classes	# Avg. Nodes	# Avg. Edges	# Node features	# Edge features
BZR-MD [25]	306	2	21.30	225.06	8	6
COX2-MD [25]	303	2	26.28	335.12	7	6
ogbg-molhiv[15]	41,127	2	25.5	27.5	9	2

Table VII: Dataset oveview with node- and edge-based features

B. Best hyperparameters on Tree-Neighboursmatch

We list the best hyperparamters for the problem "Tree-Neighboursmatch", using our dilated model with leaves. Those are hyperparameters that allowed to overfit the training set in reasonable time (up to max 24h); other configurations may lead to slower convergence. For all configurations we used Adam as optimizer and ReduceLROnPlateau (factor 0.5, patience: 1000).

Depth	Hidden size	k1	k2	Batch size	Accumulate gradients	Learning rate
2	128	1	1	128	1	1e-3
3	128	2	2	1024	1	1e-3
4	256	1	2	1024	1	1e-3
5	512	1	2	1024	1	1e-3
6	256	1	2	1024	1	1e-3
7	256	1	3	1024	2	1e-3

Table VIII: Hyperparameters on Tree-Neighboursmatch syntetic dataset

C. Hyperparameters

Layer name	# Layers (MP)	# Heads	k1 (our)	k2 (our)	Encode path length (our)	Batch size	Learning rate	h	L2	Batch norm	Act.
GCN	3 5	-	1 2 3	1 2 3	Yes No	32	1e-2	32 64	5e-4	No	relu
GIN	h=32 h=64 3 2 5 5	-	1 2 3	1 2 3	Yes No	32 128	1e-2	32 64	-	Yes	relu
GAT	3 5	4	1 2 3	1 2 3	Yes No	32 128	$1e-2$	32 64	-	No	relu
GraphSAGE	3 5	-	1 2 3	1 2 3	Yes No	32	1e-2 1e-3	32 64	-	No	relu

Layer name	Dropout	Pooling	Optimiser	Scheduler
GCN	0.0 0.5	Concat-max [16] mean	Adam	-
GIN	0.0 0.5	GIN-pooling [17] mean	Adam	Step-LR (step: 50, gamma: 0.5)
GAT	0.0 0.5	GIN-Pooling [26] mean	Adam	Step-LR (step: 50, gamma: 0.5)
GraphSAGE	0.0 0.5	Concat-max [16] mean	Adam	-

Table IX: Hyperparameter grid: for each layer we list possible values row by row