Studying the Effect of GNN Spatial Convolutions On The Embedding Space's Geometry

Claire Donnat[1] **So Won Jeong**[1]

[1]Department of Statistics, The University of Chicago, Chicago, Illinois, USA

Abstract

By recursively summing node features over entire neighborhoods, spatial graph convolution operators have been heralded as key to the success of Graph Neural Networks (GNNs). Yet, despite the multiplication of GNN methods across tasks and applications, the effect of this aggregation operation has yet to be analyzed. In fact, while most recent efforts in the GNN community have focused on optimizing the architecture of the neural network, fewer works have attempted to characterize *(a) the different classes of spatial convolution operators*, *(b) their impact on the geometry of the embedding space*, and *(c) how the choice of a particular convolution should relate to properties of the data*. In this paper, we propose to answer all three questions by dividing existing operators into two main classes (*symmetrized* vs. *row-normalized* spatial convolutions), and show how these correspond to different implicit biases on the data. Finally, we show that this convolution operator is in fact tunable, and explicit regimes in which certain choices of convolutions — and therefore, embedding geometries — might be more appropriate.

1 INTRODUCTION AND MOTIVATION

As the extension of the Deep Neural Network (DNN) machinery to the graph setting, Graph Neural Networks (GNN) offers a powerful paradigm for extending Machine Learning tools to the analysis of relational data, typically modeled through a graph [Battaglia et al., 2018, Dong et al., 2020, Hamilton et al., 2017, Kipf and Welling, 2016, Wu et al., 2020b, Zhou et al., 2020a]. The recent impressive success of GNNs across tasks and applications [Casas et al., 2019, Gaudelet et al., 2021, Gilmer et al., 2017, Li et al., 2021, Ma et al., 2019, Wu et al., 2020a] in dealing with this complex data type has been attributed to their two main components: **(a) a convolution operator** $\mathcal{C}$ (or propagation operator [Zhou et al., 2020a,b]) that aggregates information contained in the neighborhood of any given node to create neighborhood-aware embeddings; and **b) a non-linear layer** — or transformator[Zhou et al., 2020b] —, that multiplies the convolved features by a weight matrix before applying a non-linearity. All GNN parameters are typically trained in an end-to-end fashion and — as for Deep Neural Networks — have been deemed essential in creating powerful non-linear node embeddings that can be tailored to any downstream prediction task. Depending on the architecture of the network, such graph convolution blocks are then stacked and/or repeated to encode varying "receptive depths" [Frasca et al., 2020, Kipf and Welling, 2016, Wu et al., 2019]. More formally, denoting as $H^{(k)}$ the output of the k^{th} layer, GNNs can be broadly understood as a sequence of node convolutions of the form $H_u^{(k)} = \sigma(\mathcal{C}_{\mathcal{N}(u)}(H_u^{(k-1)})W_k + b_k)$, where $H_u^{(0)} = X_u$ are node u's raw features, σ is a non-linearity (e.g. ReLU), and $\mathcal{C}_{\mathcal{N}(u)}$ is the convolution operator applied to the neighborhood $\mathcal{N}(u)$ of node u. The final layer is always taken to be linear and written as:

$$H_u^{(K)} = \mathcal{C}_{\mathcal{N}(u)}(H_u^{(K-1)})W^{(K)} + b^{(K)}. \quad (1)$$

The convolution operator. Most existing theoretical analyses of GNNs differentiate two main classes of convolution operators $\mathcal{C}$[Zhou et al., 2020a]: *(i) spectral operators* [Defferrard et al., 2016, Dong et al., 2020, Gama et al., 2018a,b, Henaff et al., 2015], that build off of the eigenvectors of some version of the Graph Laplacian (defined in its unnormalized version as $L = D - A$, with A the adjacency matrix and D the degree matrix); and *(ii) spatial operators*, that recursively aggregate node features within a given neighborhood. While in practice, the dichotomy between the two is attenuated by their implementation (spectral methods usually resort to using low-order Chebychev polynomials of the Laplacian [Defferrard et al., 2016, Shuman et al., 2011, 2013], thus effectively "localizing" the signal), these two approaches have different interpre-

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

tations and implications for downstream data analysis. In this paper, we propose to focus on spatial convolutions, as popularized by the framework of Kipf et al [Kipf and Welling, 2016]. In this case, the convolution operator usually amounts to summing nodes features over entire neighborhoods $\mathcal{N}(u)$, so that the convolution becomes the function: $\mathcal{C} : \{X_v\}_{v\in\mathcal{N}(u)} \to \mathcal{C}(\{X_v\}_{v\in\mathcal{N}(u)}) = SX$, where S is a weight matrix. Kipf and Welling [2016] suggest taking S to be normalized adjacency matrix with added self-loops: $S = \hat{D}^{-1/2}\hat{A}D^{-1/2}$, where $\hat{A} = A + I$ is the adjacency matrix of the graph with self-loops, $\hat{D}$ its diagonal degree matrix, and $X \in \mathbb{R}^{n\times p}$ is the feature matrix for the n nodes. The summation is crucial in preserving permutation invariance over the neighborhood and in encoding long-range dependencies by repeatedly stacking GNN layers — thereby allowing information to percolate through the graph.

Variations on the convolution operator. While GNN methods have grown increasingly popular, the form of the spatial convolution operator seems to have received less attention from the community. Table 1 in Appendix A provides a summary of the main graph convolution operators that are currently available in the Pytorch geometric package [Fey and Lenssen, 2019] — here taken as a proxy for the most popular convolution choices. As observed from this table, all of the spatial operators rely on a (weighted) sum of neighborhood features, in line with the framework of Kipf et al [Kipf and Welling, 2016]. In fact, while GraphSage [Hamilton et al., 2017] has been attempted using alternative permutation-invariant aggregators (such as "max" pooling and an LSTM version of the convolution operator), the authors did not report any significant advantage of these methods over the simple sum. Similarly, in [Xu et al., 2018], the authors show that a simple summation is often sufficient to characterize a multiset. *Thus, without loss of (practical) generality, we restrict our study to sum-based aggregators*. In this setting, two additional variations have appeared over the recent years: *(i) attention-based spatial convolutions* [Brody et al., 2021, Veličković et al., 2017, Xie et al., 2020], which attempt to learn tailored edge weights; and more broadly, *(ii) weighted spatial convolutions* (e.g. using graphon weights, [Parada-Mayorga et al., 2021, Ruiz et al., 2020], kernels [Nikolentzos and Vazirgiannis, 2020, Feng et al., 2022] among others[Zhang et al., 2020]), that refine the adjacency matrix by endowing it with edge intensities. While efforts have thus focused on defining "the best edge weights", little attention has been put on formally analysing the convolution itself. Yet, attention-based convolutions tend to be *row-normalized*, while previous methods (e.g., GCN Kipf and Welling [2016], GINXu et al. [2018]) usually suggest weighted *symmetrized* adjacencies. Our purpose here is to show that this choice is not trivial, and translates into fundamental differences in the geometry of the embedding space.

Prior work: studying the effect of the Convolution operator. The literature focusing on understanding the mechanisms behind the success of Graph Neural Networks has considerably expanded over the past few years. Most notably, an important body of work has focused on understanding the role of this convolution operator in phenomena such as oversmoothing [Oono and Suzuki, 2019, Cai and Wang, 2020, Chen et al., 2020] and oversquashing[Alon and Yahav, 2020, Topping et al., 2021]. Most of these analyses are led from a spectral perspective, relating the behavior of the embeddings to the convolution operator's eigenvalues. However, to the best of our knowledge, none of these approaches have specifically focused on understanding the effect of the convolution operator on the organization of the underlying embedding space, nor has attempted to tie these properties to any *topological features*. While methods have tried to embed nodes in specific manifolds with desirable geometries (e.g., hyperbolic [Liu et al., 2019, Law, 2021]), no work seems to have studied the geometry induced by the convolution itself. We posit that such considerations are nonetheless fundamental in our understanding of GNNs and their stability.

Contributions. The objective of this paper is to answer three questions: *(a) how does the choice of a particular convolution operator affect the organisation of the embedding space*, *(b) how does it relate to the original properties (i.e. node features, graph distances or topological attributes)*, and *(c) what is the most appropriate convolution operator for a given dataset?* We will attempt and answer all three questions by studying two larger families of row-normalized and symmetrized convolution operators (parametrized by the variables $\alpha \in [0, 1]$ and $\beta \in \mathbb{R}^+$), allowing us to show how the convolution operator itself is in fact tunable. In particular, we will show different values of α and β impact the organization of the latent space (Section 3) and the inherent geometry of the embeddings (Section 4). Finally, we will characterize regimes in which certain choices of operators might be more relevant than others.

2 DEFINING A FAMILY OF CONVOLUTION OPERATORS

To analyse the impact of the convolution operator on the embedding geometry, we define two main families of spatial convolutions:

a. Symmetrized Convolutions Operators, defined as the family of operators of the form:

$$\mathcal{F} = \big\{M_{\alpha,\beta} = D_\beta^{-\alpha}(A+\beta I)D_\beta^{-\alpha}\big|\alpha \in [0,1], \\ \beta \in \mathbb{R}^+, D_\beta = \mathrm{diag}\big((A+\beta I)\mathbb{1}\big)\big\}. \quad (2)$$

Here, D_β is the degree matrix associated with the β-augmented adjacency matrix $A + \beta I$. Note that this family is a generalization of the traditional GCN convolution $S_{GCN} = \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$, which corresponds here to a choice of $\alpha = 0.5$ and $\beta = 1$. The choice $\alpha = 0.5$ and

$\beta = 2$ also feature amongst the default implementations in PyTorch Geometric [Fey and Lenssen, 2019]. Similarly, the convolution operator indexed by $\alpha = 0, \beta = 1$ corresponds to a version of the GIN convolution [Xu et al., 2018], and more generally, to the sum-based message-passing versions of GNNs [Battaglia et al., 2018].

b. Row-normalized Convolution operators, which we define as the general family of the form:

$$\mathcal{M} = \big\{ D^{-1}_{(\alpha,\beta)} M_{\alpha,\beta} \big| D_{(\alpha,\beta)} = \operatorname{diag}\big(M_{\alpha,\beta}\mathbb{1}\big), \quad M_{\alpha,\beta} \in \mathcal{F} \big\}. \tag{3}$$

This family encompasses a number of operators, such as the sum-based convolution deployed in GraphSage[Hamilton et al., 2017] — and, to some extent, that of GAT [Veličković et al., 2017], which considers row-normalized convolutions of a "learned", modified version of the adjacency matrix.

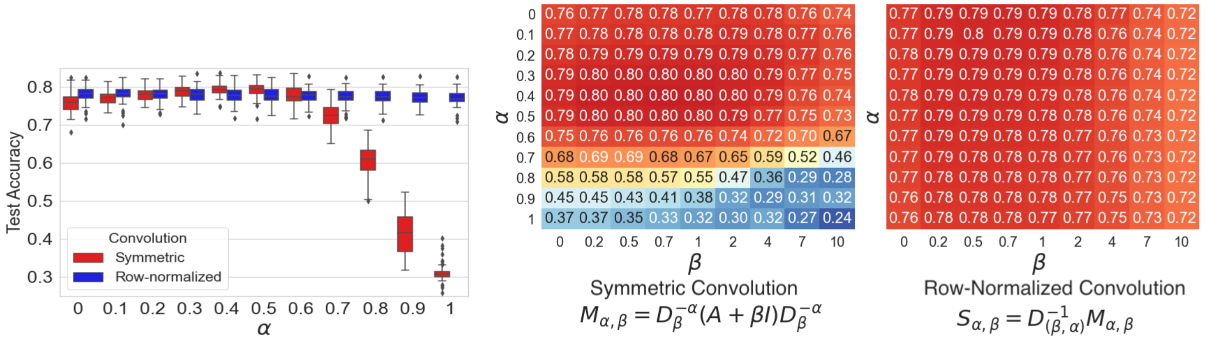
For both families of convolution operators, the parameter α impacts the weights assigned to nodes as a function of their degree: as α increases, high-degree nodes are increasingly penalised, so that their contribution to neighboring node embeddings decreases (relative to lower-degree nodes). On the other hand, β can be interpreted as capturing the amount of "innovation" or relevant information that the source node brings to the embedding with respect to its neighborhood. In particular, for high values of β, the source node's contribution to the node outweighs that of the neighborhood, so that the embedding becomes essentially identical to its source. Consequently, β allows us to interpolate between the traditional GNN regime ($\beta = 1$) and the MLP setting. Table 1 in Appendix A lists a number of popular GNN convolutions, along with their associated families and parameters — thereby highlighting the ubiquity of this framework.

Empirical consequences of a choice of operator. While seemingly specious, the distinction between these two families corresponds in fact to different assumptions on the nature of the data. Consider the (potentially weighted) adjacency matrix with self-loops as a similarity matrix. GNN row-normalized convolutions bear a striking resemblance with the diffusion maps suggested by Coifman and Lafon [2006] for embedding nodes in the "featureless" case where $X = I$. Diffusion maps are embeddings provided by the eigenvectors of the matrix: $M = D^{-1}_{\alpha,\beta} M_{\alpha,\beta}, \quad M_{\alpha,\beta} \in \mathcal{F}$ for an appropriate choice of β. Coifman and Lafon [2006] show indeed more generally that the eigenvectors of the matrix $M^t, t \in \mathbb{R}^+$ allow to recover the structure of manifold underlying the graph at larger and larger scales. The stacking of the different GNN convolutions without non-linearities (e.g. Wu et al. [2019]) can be compared to a variation of the diffusion process proposed by Coifman and Lafon [2006]: in the case of GNNs, $t \in \mathbb{N}$ is discrete and corresponds to the depth of the network. This connection is interesting. Coifman and Lafon [2006] emphasized indeed the importance of the choice of the α: if the data density is assumed to be uniform, then a choice of $\alpha = 0$ ensures that the operator is an unbiased estimate of the Laplace-Beltrami operator. Conversely, $\alpha = 1$ is better suited for manifold estimation with non-uniform sampling densities. While GNNs consider embeddings of both the graph and its node features, we posit that α might have a similar effect on the estimation procedure. This comparison implicitly assumes that the data lives on some smooth Riemannian manifold, where the degree or centrality of a node corresponds to an assumption on the sampling distribution over $\mathcal{M}$: nodes with high degree correspond to "well-sampled" areas of the manifold. Therefore, in this setting, it is intuitively possible to get an accurate representation of the local information by averaging neighbourhood features: the higher the degree, the higher the amount of certainty around the node's value. By contrast, symmetrized embeddings are weighted sums — but not convex combinations — of neighbours. Here, the sum simply plays the role of a permutation-invariant aggregation operator[Hamilton, 2020], and as we will see, is able to encode topological features (e.g. structural roles [Donnat et al., 2018]).

Experiments. Finally, to motivate our study before diving into more theoretical considerations, we propose to highlight the impact of the choice using standard benchmarks in the literature (we refer the reader to Appendix E for an overview of the properties of these datasets)[1]. Figure 1b highlights the impact of the value of α and β on the classification accuracy for Cora. Note here that we are not suggesting the use of any particular tuple (α, β) for Cora, but simply highlighting its influence on the performance of the algorithm. In particular, for the symmetric operator, the value of α is the main driver of the difference in performance. By contrast, the choice of β affects less the performance of the GNN— unless β becomes too big and outweighs the rest of the neighbors. This effect might be due to the significant level of homophily in Cora: in this setting, the source node's feature vector is fairly redundant with that of its neighbours. However, we show in Appendix E additional examples where the impact of β is much more substantial. Most strikingly, the choice of α seems to have a significant effect on the performance of the symmetrized GNN, with a phase transition at $\alpha = 0.5$: for values of α greater than this threshold, the performance drops quite substantially. Noticeably, in the "poor" performance region, the interaction between α and β is more marked: choosing the low value of β (i.e., $\beta = 0$) seems to mitigate the decrease in performance.

We emphasize here that the scope of this paper is not to suggest another convolution operator that would achieve state-of-the-art results. Rather, through this series of experiments, we hope to have convinced the reader that, empirically, the choice of convolution is important and can help gain up to almost 7% accuracy on traditional GNN approaches, with no modifications to the architecture of the network whatso-

[1] The code for the experiments can be found here



(a) Accuracy as a function of α ($\beta = 1$).

(b) Cora dataset: test accuracy as a function of α and β

Figure 1. Results for Cora for our family of convolutions defined in Eq.2 and Eq.3 (50 independent experiments, selecting a random training set and test set). Note the strong dependency of the results on both α and β for the normalized convolution.

Dataset	Alpha Convolution Type	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Cora	Symmetric	77.02±2.13	77.85±1.7	78.91±1.78	**79.37±1.78**	79.23±2	77.63±2.53	72.71±3.43	60.67±4.09	41.59±5.71	31.19±2.59
	Row-Normalized	**78.12±2.27**	78.08±2.01	77.97±2.16	77.83±2.1	77.81±2.07	77.73±1.91	77.3±2.24	77.33±2.21	77.24±2.01	77.1±2.2
pubMed	Symmetric	75.41±2.51	76.09±2.38	76.63±2.51	**76.79±2.48**	75.4±3.81	68.27±6.82	54.62±8.52	44.11±5.36	40.57±2.13	39.9±2.09
	Row-Normalized	74.42±3.82	74.62±3.76	74.74±3.65	**74.75±3.44**	74.68±3.45	74.48±3.44	74.28±3.38	73.92±3.36	73.73±3.36	73.42±3.44
Citeseer	Symmetric	65.06±2.74	65.5±2.57	66.51±2.49	**67.59±2.59**	67.45±2.56	67.32±3.04	65.53±3.58	60.77±4.17	51.41±4.86	38.2±5.07
	Row-Normalized	66.69±2.87	66.75±2.99	66.79±2.69	66.99±3.07	66.74±2.71	66.91±2.84	**67.02±3.03**	66.98±2.92	66.93±2.92	66.96±2.97
Coauthor CS	Symmetric	**93.02±0.27**	92.93±0.25	92.9±0.21	92.32±0.26	88.66±0.39	77.88±0.89	52.34±1.93	24.91±0.48	22.63±0.29	22.63±0.29
	Row-Normalized	88.97±0.4	89.4±0.43	89.74±0.41	90.01±0.45	90.13±0.51	90.21±0.5	**90.29±0.5**	90.26±0.47	90.3±0.44	90.24±0.42
Amazon Photos	Symmetric	58.77±18.34	88.67±1.62	**90.09±1.02**	89.18±1.07	83.23±1.64	37.12±1.4	33.17±0.93	32±0.91	29.05±0.94	27.47±0.86
	Row-Normalized	82.09±1.34	83.75±1.13	84.97±1.34	86.15±1.26	86.75±1.44	87.33±1.18	88.01±1.11	87.9±1.2	**88.05±0.99**	87.46±1.3
Actor	Symmetric	27.06±0.61	27.43±0.64	27.74±0.61	28.27±0.54	28.58±0.57	28.75±0.52	**28.9±0.56**	28.83±0.6	28.51±0.67	28.33±0.64
	Row-Normalized	29.42±0.73	29.97±0.54	30.33±0.65	30.71±0.71	31±0.83	31.26±0.71	31.37±0.72	31.53±0.6	31.58±0.6	**31.63±0.6**
Cornell	Symmetric	51.04±4.08	51.01±3.25	51.04±2.91	51.28±3.08	51.25±3.12	52.29±3.06	53.65±3.21	54.38±4.17	**54.55±3.97**	54.55±4.65
	Row-Normalized	63.78±4.39	64.13±4.8	63.96±4.53	64.65±4.51	64.83±4.74	65.52±4.65	65.87±4.72	65.94±4.6	**66.46±4.5**	66.39±4.61
Wisconsin	Symmetric	48.03±3.69	49.39±3.5	50.29±3.04	51.09±3.15	**51.79±3.1**	51.6±3.65	50.8±4.59	49.92±5.04	49.07±5.16	48.21±4.75
	Row-Normalized	50.24±4.88	51.04±5.38	51.28±5.77	52.03±6.07	52.91±6.32	53.6±5.82	54.4±6.01	55.28±6.06	55.76±6.01	**56.19±5.84**
PATTERN	Symmetric	77.88±11.43	79.85±7.07	79.55±9.32	81.67±6.49	83.48±4.19	**84.39±3.03**	84.09±2.88	84.09±2.88	84.09±2.88	84.09±2.88
	Row-Normalized	79.7±7.77	81.36±5.44	81.67±5.37	83.03±3.56	83.48±2.98	83.79±3.28	**84.24±2.78**	83.33±4.04	82.58±3.13	83.48±2.62
CLUSTER	Symmetric	36.14±15.88	34.29±10.24	39.86±12.84	32.71±11.16	37.29±14.56	38.14±12.09	38.29±14.69	**42.43±6.43**	31.57±8.87	23.71±4
	Row-Normalized	**45.29±11.74**	37.71±11.39	41.43±9.52	37.71±9.69	38.14±11.19	37.86±10.91	32.43±10.01	33.29±9.92	33.29±12.25	28.14±11.61
WikiCS	Symmetric	26.38±3.18	30.83±3.21	51.85±5.45	64.36±3.96	**66.02±3.87**	59.54±2.37	51.9±4.56	39.53±2.61	33.64±3.56	31.89±1.76
	Row-Normalized	52.14±4.28	58.89±2.14	60.69±2.51	62.56±5	64.33±2.81	66.68±3.59	68.91±2.15	69.01±2.19	68.08±3.37	**69.23±3.98**
OGBN-arxiv	Symmetric	16.12±0.05	16.4±0.39	17.14±1.03	16.85±1.7	16.14±0.03	16.44±0.2	16.95±0.08	17.18±0.11	**17.19±0.08**	16.81±0.19
	Row-Normalized	**17.35±0.2**	17.07±0.33	16.9±0.08	16.69±0.12	16.57±0.12	16.36±0.27	16.18±0.1	16.14±0.03	16.14±0.03	16.14±0.03

Table 1. Results(accuracy) of node classification task for 12 benchmark datasets. The number of experiments differs by each dataset. Batch normalization has been applied to *PATTERN*, *CLUSTER*, *WikiCS*. Details for the experiments are provided in Table 3 in Appendix E.

ever. Motivated by these observations, the rest of this paper focuses on analysing these convolution operators, and their impact on the organization of the embedding space.

3 GEOMETRY OF GNN EMBEDDINGS IN LATENT SPACE

In this section, we analyse the effect of the convolution operator on the global organization of the latent space. Our objective is to (a) characterize the implicit constraints that these operators put on the geometry (in particular, that embeddings concentrate differently according to their degree and choice of the operator), and (b) identify the downstream consequences in terms of performance. This discussion is driven by considerations on nodes' topological characteristics — rather than spectral arguments.

3.1 SYMMETRIC CONVOLUTIONS

We begin our study of the "absolute" latent geometry of our embeddings with the family of symmetric convolution $\mathcal{M}_{\alpha,\beta}$ (see Equation 2). For each layer K, the embedding is defined as:

$$H^{(K)} = S\sigma(H^{(K-1)}W + b) = \sum_{v \in \tilde{\mathcal{N}}(u)} \frac{A_{uv}}{(d_u + \beta)^{\alpha}(d_v + \beta)^{\alpha}} Z_{v\cdot}$$

where $\tilde{\mathcal{N}}(u) = \mathcal{N}(u) \cup \{u\}$ denotes the neighborhood of node u, A_{uv} is the (possibly weighted) adjacency matrix, with diagonal equal to β, and $Z_{v\cdot} = \sigma(H_{v\cdot}^{(K-1)}W + b)$.

As a first step to study of the effect of the choice of convolution operator on the latent embedding space, we propose the following lemma.

Lemma 3.1. *For any node u, the effect of the convolution* *can be characterized as follows:*

$$||SZ||_2 \leq ||Z||_{2,\infty}\Big((d_u+\beta)^{1-2\alpha} - \alpha\frac{\bar{\Delta}_u}{(d_u+\beta)^{2\alpha}} + \frac{\alpha(\alpha+1)M}{2}\frac{\overline{\Delta^2}_u}{(d_u+\beta)^{1+2\alpha}}\Big) \tag{4}$$

where $\bar{\Delta}_u$ (respectively $\overline{\Delta^2}_u$) are the weighted averages of the degree differences (respectively, squared degree differences):

$$\bar{\Delta}_u = \frac{\sum_{v\in\tilde{\mathcal{N}}(u)} A_{uv}(d_v - d_u)}{d_u+\beta}, \overline{\Delta^2}_u = \frac{\sum_{v\in\tilde{\mathcal{N}}(u)} A_{uv}(d_v - d_u)^2}{d_u+\beta}$$

In this equation, $||Z||_{2,\infty} = \max_v ||Z_v||_2$, and $M = \frac{d_{\max}+\beta}{\beta+1})^2$, where $d_{\max}$ denote the maximal degree of the nodes in the network.

Proof The proof is simple (see Appendix B), and relies on the triangle inequality coupled with a MacLaurin expansion of the function $d_v \to \frac{1}{(d_u+\beta)^\alpha(d_v+\beta)^\alpha}$. □

Note that this bound is not necessarily tight. In particular, the proof relies on an application of the triangular inequality, along with Hölder's inequality to separate the convolution from the embeddings. However, while potentially crude, this bound already allows us to shed more light on the behaviour of the embedding as a function of the parameters α, β, and their topology. In particular, this bound allows to highlight:

(a) The role of α. The leading term in inequality 4 is $d_u^{1-2\alpha}$, and offers a first explanation for the change of phase we have observed in some of our experiments in the previous section. For values of $\alpha < 0.5$, this term is an increasing function of the degree: after even a single convolution, nodes with a small degree have less leeway to spread, and will generally remain close to the origin; High-degree nodes, on the other hand, will enjoy greater variance after each convolution and be able to spread to greater radii. Conversely, if $\alpha > 0.5$, the upper bound decreases for nodes with high degree — forcing them to concentrate around the origin. The parameter α, therefore, controls the "attraction" of nodes towards the origin as a function of their degree.

(b) The effect of β. The coefficient β, on the other hand, acts as added mass to the degree d_u and can be understood as the "strength" of the attraction: for $\alpha > 0.5$, the attraction of high-degree nodes to the origin is an increasing function of β. Conversely, for $\alpha < 0.5$, the repulsion of the nodes from the origin is an increasing function of β.

(c) The influence of the surrounding topology. As previously exhibited, the node degree plays a defining role in the variance of the embedding. The bound also exhibits a dependency on the neighborhood topology through the terms $\bar{\Delta}_u$ and $\bar{\Delta}^2{}_u$. Therefore, the more topologically homogeneous the neighborhood, the lesser the variance.

Consequences. The previous observations yield two main conclusions. First, the choice of the convolution vector drives the density of the embedding space: values of β and α allow the embedding space to expand or contract around the origin, depending on the node degree.
The second consequence pertains to the accuracy of the recovery. Since the last layer of GNN is a linear classifier, we can use known results from statistical theory about the influence of the different points on the performance of the algorithm. In particular, in linear regression, it is known that high-leverage points (that is, points with "extreme" predictor values") are more likely to be highly influential points Weisberg [2005] (the same follows for generalized linear models, with some nuances). As such, by preventing high-degree nodes (respectively low-degree) from taking on extreme embedding value and concentrating them around the origin, the convolution operator implicitly limits the amount of trust, or leverage, that these points may have. We summarize these observations in the following lemma.

Lemma 3.2 (Effect of symmetric convolutions on node embeddings)**.** *In networks with non-homogeneous degree distributions, the exponent α constrains the leverage associated with each of the embeddings as a function of their degree and local topology:*

- **For values of $\alpha > 0.5$:** *High-degree nodes are constrained to concentrate around the origin, allowing the performance of the algorithm to be driven by more peripheral (low-degree) nodes.*
- **For values of $\alpha < 0.5$:** *Low-degree nodes need to lie closer to the origin than their high-degree counterparts, thereby allowing high-degree nodes to have higher leverage and potentially become more influential.*

Experiments. To illustrate these bounds and check their validity, we perform a set of synthetic experiments. We generate a set of four cliques on 20 nodes (the "hubs"). To each node in each clique, we add a link to a sparse Barabasi-Albert network on 10 nodes ('the periphery'), with parameter $m = 1$, so that the average degree of the periphery is low (≈ 1). The peripheries are endowed with the same class label as their associated hub. Finally, we ensure that the network is connected by randomly connecting the hubs together (one new random link per hub). To generate node features, we take the first $k = 4$ features to be the one-hot label vector, to which we concatenate 16 additional "dummy features"(random Gaussians). We finally add Gaussian noise with scale $\sigma^2 = 4$ entrywise: the result is a feature vector that is only weakly indicative of the class. The trained embeddings are presented in Figure 2. Note the lack of separability of the different classes based on their raw feature vectors, as captured by the PCA plot in the left column. As expected from our bounds, we observe an inversion of the geometry around the origin as α increases: high-degree nodes shift from the outskirts of the plot to being concentrated around the origin as α increases. We also refer the reader to Appendix E.3.2, in which we also verify these phenomena in standard datasets by providing PCA and UMAP plots of the corresponding node embeddings.

In order to test lemma 3.1, we modify this experiment slightly, and now let the variance of the noise depend on the degree d_u of the node: $\sigma_u^2 = e^{3(-1.5+\log(d_u))}$. This means that low-degree nodes here have very small variance $\sigma < 1$, while high-degree noise is extremely noisy $\sigma \approx 9$. Consequently, we expect that the geometries in which the high-degree nodes are placed on the outskirts (and have more leverage), and low-degree nodes are constrained to lie close to the origin will perform poorly. Conversely, we flip this scenario (we choose $\sigma_u^2 = e^{3(-1.5+\log(d_{(n-u)})}$ where, if d_u has rank $rk(u)$, $d_{(n-rk(u))}$ is the degree of the $n-rk(u)$ largest node), and expect the opposite phenomenon. The results are shown in Figure3(a), and are well aligned with our expectations.

3.2 ROW-NORMALIZED CONVOLUTIONS

Let us now turn to the case of row-normalized convolutions. In this case, the convolutions write as:

$$\begin{aligned} SX &= \frac{\frac{1}{(d_u+\beta)^\alpha} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{1}{(d_v+\beta)^\alpha} X_v}{\sum_{v \in \mathcal{N}(u)} \frac{1}{(d_u+\beta)^\alpha} \frac{1}{(d_v+\beta)^\alpha}} \\ &= \frac{1}{\sum_{v \in \mathcal{N}(u)} \frac{1}{(d_v+\beta)^\alpha}} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{1}{(d_v+\beta)^\alpha} X_v \end{aligned}$$

The embedding now lies within the convex hull of its neighbors, whose contributions are inversely proportional to their degrees. This reduces the sensitivity of variance of the node embeddings to their degree. The decay of a neighbor's contribution is fact an increasing function of α. The latter can be compared to a form of attention that effectively filters out nodes with high-degree: here the discounting procedure is not learned but imposed ahead of time. Appendix E and Figure3(b) show the result of the same experiments as in the last subsection. As in the previous part, the results are less dependent on the value of α.

4 INHERENT EMBEDDING GEOMETRY

We now turn to the analysis of the evolution of the relative distances between embedding points. Graph Neural Networks can indeed be understood as a data integration method: the adjacency matrix and the node features provide two complementary views of the data, and GNNs provide a convenient way of combining this information to create informative embeddings. The amount to which these embeddings depend on one side of the data (i.e. graph vs node features) remains to be determined.

Let us try and study the effect of the different distances (graph and topology) through two toy examples controlling for feature similarity and node similarity separately.

Toy example 1: Identical Topologies, Different features. Consider two nodes u and v have structurally similar neighborhoods (i.e., there exists a mapping ϕ that transforms each node in the neighborhood of v into its corresponding one in the neighborhood of u, see Appendix C (Figure 1), but whose feature vectors are different. Mathematically, we write:

$$\forall j \in N(v), \quad X_{j\cdot} = X_{\phi(j)\cdot} + \epsilon, \qquad \epsilon_{jk} \overset{\text{i.i.d}}{\sim} N(0, \sigma^2).$$

Lemma 4.1. *For symmetric convolution, with probability at least* $1-\delta$*, with* M *as in 3.1, we have:*

$$\begin{aligned} ||H_u - H_{u'}||^2 \leq \mu + 2\sqrt{2}\sigma ||W||_2 (d_u+\beta)^{1-2\alpha} \\ \times \sqrt{1 + 2\alpha|\overline{\Delta}_u| + \alpha(2\alpha+1)\frac{M\overline{\Delta^2}_u}{d_u}} \log(1/\delta) \end{aligned}$$

where $\mu = \sigma^2 ||W||^2 (d_u+\beta)^{2-4\alpha}\Big(1 + 2\alpha|\overline{\Delta}_u| + \alpha(2\alpha + 1)M\frac{\overline{\Delta^2}_u}{d_u}\Big)$.

Conversely, for row-normalized embeddings:

$$\begin{aligned} ||H_u - H_{u'}||^2 \leq &\mu + 2\sqrt{2}\sigma ||W||_2 \\ &+ \Big(\sum_{v \in \tilde{\mathcal{N}}(u)} \frac{1}{(d_v+\beta)^{2\alpha}}\Big)^{1/2} \log(1/\delta) \end{aligned}$$

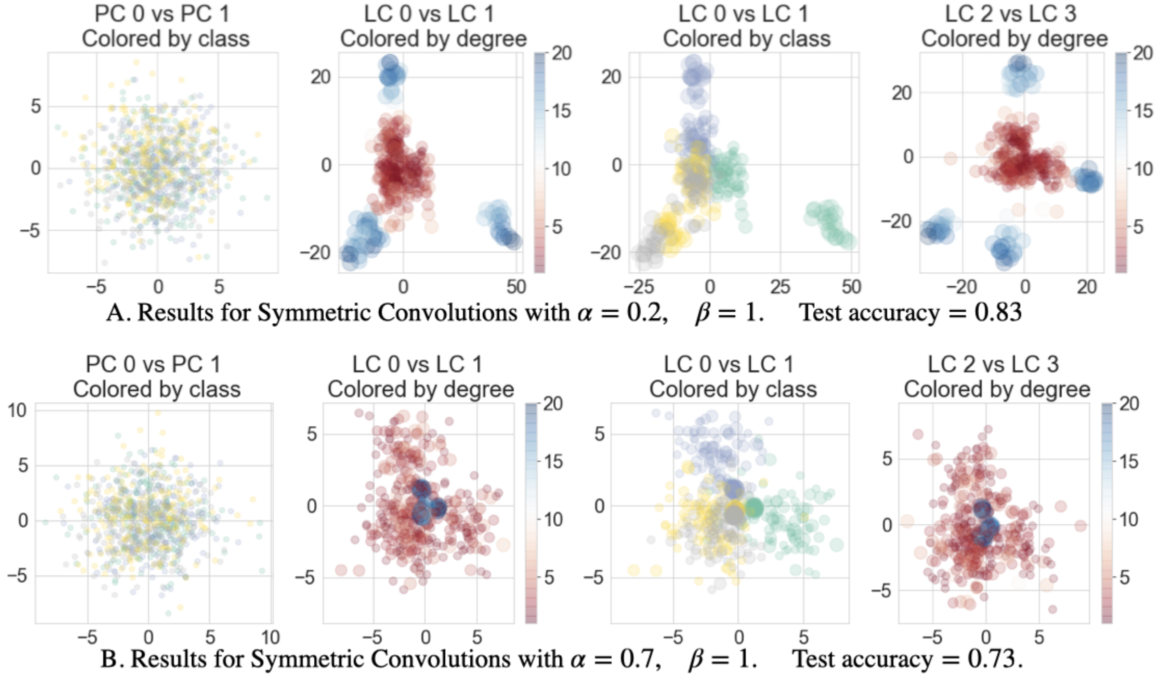
where $\mu = \frac{\sigma^2 \|W\|^2}{\sum_{v \in \tilde{\mathcal{N}}(u)} (d_v+\beta)^{-\alpha}} \frac{1}{1+\beta} \leq \beta\sigma^2 \|W\|^2$ *for* $\beta \geq 1$.

The proof is in Appendix C. Symmetric embeddings will thus shrink distances between nodes depending on their topology, so that the cluster density is a function of the node degree. The impact of the topology is expected to be more marked at the extremities of the spectrum of α. For row-normalized embeddings however, the degree of the node does not affect the distance between embeddings as much: row-normalized embeddings encode attributes, rather than topological information.

Toy example 2: Identical Features, structurally different neighbourhoods. Conversely, u and v have radically different neighborhoods from a topological perspective, but have similar features:

$$\forall j \in \tilde{\mathcal{N}}(u) \cup \tilde{\mathcal{N}}(u'), X_j = \bar{X} + \epsilon$$

In this case, for row-normalized embeddings, the difference will be 0: the embeddings are therefore more sensitive to the node feature values than the symmetric convolution. Conversely, for symmetric convolutions, we can also show (see Appendix C) that the leading term for the difference is of the order of $(d_u+\beta)^{1-2\alpha} - (d_{u'}+\beta)^{1-2\alpha}$ — and is, therefore, a decreasing function of α.



A. Results for Symmetric Convolutions with $\alpha = 0.2$, $\beta = 1$. Test accuracy = 0.83

B. Results for Symmetric Convolutions with $\alpha = 0.7$, $\beta = 1$. Test accuracy = 0.73.

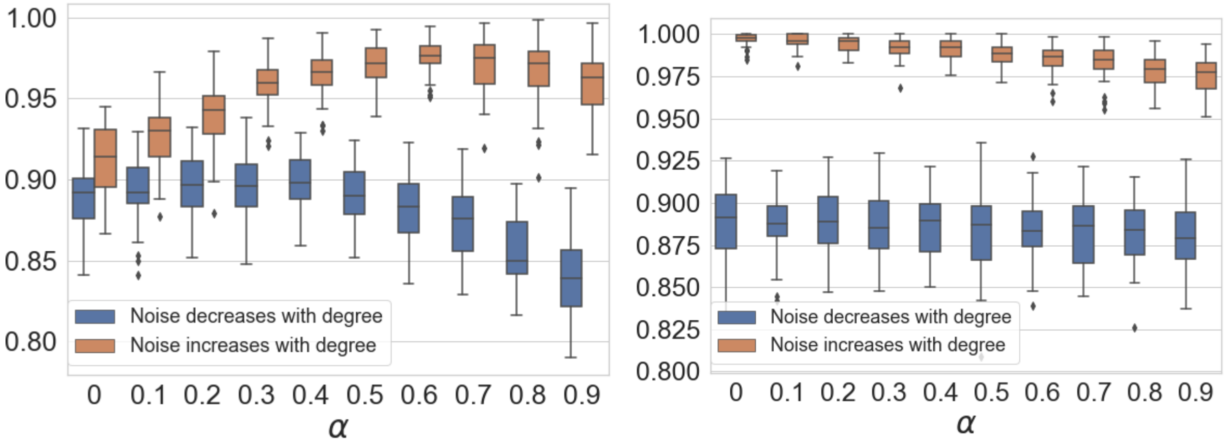
Figure 2. Symmetric Embeddings, plotted using the first two principal components (left), and the raw latent embeddings (or 'latent components' (LC), shown in the right three plots on each row). Note the inversion: the high-degree nodes migrate from the periphery of the latent space ($\alpha = 0.2$) to the origin ($\alpha = 0.7$). See Appendix E for the equivalent plot for row-normalized Embeddings.

Experiments. We illustrate these results by running a set of final experiments. We generate a structurally equivalent networks (see smaller replica in Figure 1a in Appendix C), and evaluate the distance between untrained embeddings (using a 2-layer GCN architecture). The mean distance over 100 experiments is presented in Appendix C in Figure 1b, and a visualization of the latent space for symmetric convolutions is shown in Appendix C, Figure 1c. As expected, the density of the cluster of structurally equivalent high-degree nodes (cliques on 40 nodes), in grey varies as a function of α. In Appendix E.3.3, we also provide visualization of the interplay between node features and topologies on a subset of real datasets, using the Gromov Wasserstein distance as a way of measuring the distance of the embedding space with the original characteristics of the dataset (features and adjacency matrix).

Toy Example 3: Degree corrected Stochastic SBM. We conclude this section by considering a specific family of graphs: the degree-corrected Stochastic Block Model Karrer and Newman [2011] on two classes of equal size n. Let each node have class $Z_i \in \{1, 2\}$, and denote $X_i = \mu^{(Z_i)} + \epsilon_i$ its attributes. According to the DC-SBM model, each edge in the network is sampled according to a Bernouilli distribution: $A_{ij} \sim \text{Bernouilli}(\theta_i \theta_j \omega_{Z_i Z_j})$, where θ_i is a popularity parameter such that, for each group g: $\sum_{i=1}^{n} \theta_i 1_{Z_i = g} = n$, where ω_{ij} is the parameter of the model corresponding to the probability of connection between group i and j. Note that, under this model, the expected number of edges from community (i) to (j) is simply $m_{ij} = n^2 \omega_{ij}$. Therefore, picking $\forall i, \theta_i = 1$ corresponds to the traditional stochastic block model.

In this case, it is possible to show (see Appendix D) that the mean of the symmetric embeddings is directly proportional to their popularity parameter $\theta_i^{1-\alpha}$. Consequently, for $\alpha = 1$, the leading term is independent of θ_i. Reciprocally, for $\alpha = 0$, the embedding is directly proportional to θ_i. This means that the embeddings will on average have a norm that is proportional to their popularity: the embedding space thus capture the "popularity" of the embeddings through their degree.

To see this, we provide the following example. Consider a DC-SBM graph on 300 nodes with two classes, with connectivity parameters $\omega_{11} = \omega_{22} = 0.1$ and $\omega_{11} = \omega_{22} = 0.005$. The features here are taken to be multivariate normal with $\mu^{(1)} = 2, \mu^{(2)} = -2$ and standard deviation equal to 4. We generate the θ_i for each group from a lognormal distribution, with mean 0 and standard deviation 1 (see details in Appendix D). In the results (Fig 4 and Appendix D), we observe as predicted the high dependency of the embedding on the "popularity" parameter for low values of α.



(a) Results for symmetric convolutions.

(b) Results for row-normalized convolutions.

Figure 3. Results for 50 independent trials of our second experiment using symmetric (left) and row-normalized (right) convolutions. Note that the accuracy of the symmetric convolutions increases α increases when the noise is higher in high-degree nodes, and decreases in the opposite scenario.

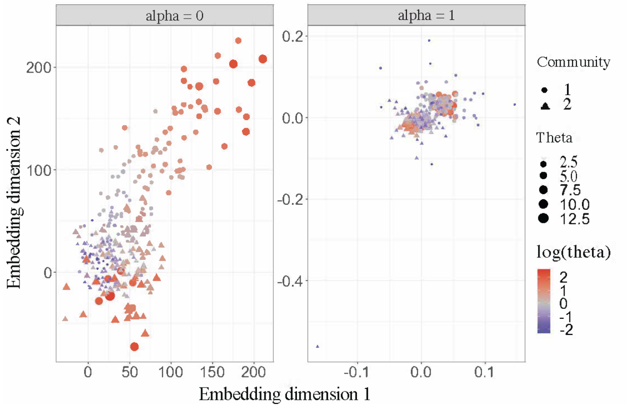


Figure 4. Embeddings after one convolution for two different values of $\alpha \in \{0, 1\}$, and $\beta = 1$. Note how the value of the popularity parameter θ drives the geometry of the embedding space when $\alpha = 0$.

5 DISCUSSION

To summarize, this paper has tried to explicit two main phenomena: (a) the dependency of the variance of the embeddings on the degree, choice of convolution operator and parameter α; (b) The higher sensitivity of the symmetric embedding distances to topological features compared to that of the row-normalized one. We now conclude by discussing the practical impact of these observations. Consider the two following use cases:

(a) Learning user embeddings in a social network, such as for instance in Zitnik et al. [2018]. Here, the degree can be considered as an additional dimension of information: users with high degree might be more popular or sociable, and therefore more alike to one another. This information should be reflected in the embedding space. In the first, a symmetric embedding — which is typically more sensitive to distance between topological features — might be more suitable to the task. Moreover, using our results on the degree-corrected stochastic block model, the lower the α chosen, the higher the potential emphasis on the degree.

(b) Learning drug embeddings in biological network (e.g. Zitnik et al. [2018]) In this case, the degree of the node might not necessarily be as informative: some drugs may have been on the market for longer, and/or their mechanisms of actions are better understood. In that case, the features of a drug's neighbors might be informative, but not necessarily their degree. Conversely, a row-normalized embedding might prove a better choice. Alternatively, the symmetric convolutions with $\alpha = 1$ would mitigate the effect of the sampling density. We note however that the rapid contraction of the embeddings towards 0 makes it difficult for the GNN to learn informative embeddings.

6 CONCLUSION

In conclusion, in this paper, we have shown that the choice of the convolution operator has fundamental consequences on the geometry of the embedding space: symmetric convolutions are generally more sensitive to the topology, and encode it in the embedding. In that case, the choice of α amounts to selecting "who to trust": high-values of α push high-degree nodes towards the origin, thereby limiting their leverage. Conversely, row-normalized are more limited in the amount of topological information that they carry, and convolutions are more robust to the choice of α — this is probably a better choice when the data is assumed to be sampled from a manifold (e.g. point cloud data). Our analysis

— which we hope to be insightful — has room for further improvement. Our reasoning relies on upper bounds which, while providing intuition, are not extremely tight, and could be complemented with lower bounds to fully characterize the behavior of the geometry. All experiments resort to using GCN types of architectures. However, we believe that the intuition and guidelines that we derived from this analysis will nonetheless hold for other types of architectures.

Acknowledgements

The authors are grateful for the support of the University of Chicago's Research Computing Center for assistance with the calculations carried out in this work, as well as for a generous award from Facebook Research (2021 Proposal for Statistics for Improving Insights, Models, and Decisions) that supported this research.

References

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

Sergio Casas, Cole Gulino, Renjie Liao, and Raquel Urtasun. Spatially-aware graph neural networks for relational behavior forecasting from sensor data. *arXiv preprint arXiv:1910.08233*, 2019.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445, 2020.

Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*, 2016.

Xiaowen Dong, Dorina Thanou, Laura Toni, Michael Bronstein, and Pascal Frossard. Graph signal processing for machine learning: A review and new perspectives. *IEEE Signal processing magazine*, 37(6):117–127, 2020.

Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1320–1329, 2018.

Aosong Feng, Chenyu You, Shiqiang Wang, and Leandros Tassiulas. Kergnns: Interpretable graph neural networks with graph kernels. *arXiv preprint arXiv:2201.00491*, 2022.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020.

Fernando Gama, Antonio G Marques, Geert Leus, and Alejandro Ribeiro. Convolutional neural network architectures for signals supported on graphs. *IEEE Transactions on Signal Processing*, 67(4):1034–1049, 2018a.

Fernando Gama, Alejandro Ribeiro, and Joan Bruna. Diffusion scattering transforms on graphs. *arXiv preprint arXiv:1806.08829*, 2018b.

Thomas Gaudelet, Ben Day, Arian R Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy BR Hayter, Richard Vickers, Charles Roberts, Jian Tang, et al. Utilizing graph machine learning within drug discovery and development. *Briefings in bioinformatics*, 22(6):bbab159, 2021.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017.

William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical review E*, 83(1):016107, 2011.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Marc Law. Ultrahyperbolic neural networks. *Advances in Neural Information Processing Systems*, 34:22058–22069, 2021.

Michelle M Li, Kexin Huang, and Marinka Zitnik. Representation learning for networks in biology and medicine: Advancements, challenges, and opportunities. *arXiv preprint arXiv:2104.04883*, 2021.

Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Tengfei Ma, Junyuan Shang, Cao Xiao, and Jimeng Sun. Genn: predicting correlated drug-drug interactions with graph energy neural networks. *arXiv preprint arXiv:1910.02107*, 2019.

Giannis Nikolentzos and Michalis Vazirgiannis. Random walk graph neural networks. *Advances in Neural Information Processing Systems*, 33:16211–16222, 2020.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

Alejandro Parada-Mayorga, Luana Ruiz, and Alejandro Ribeiro. Graphon pooling in graph neural networks. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 860–864. IEEE, 2021.

Luana Ruiz, Luiz Chamon, and Alejandro Ribeiro. Graphon neural networks and the transferability of graph neural networks. *Advances in Neural Information Processing Systems*, 33:1702–1712, 2020.

David I Shuman, Pierre Vandergheynst, and Pascal Frossard. Chebyshev polynomial approximation for distributed signal processing. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–8. IEEE, 2011.

David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 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, 2013.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021.

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

Sanford Weisberg. *Applied linear regression*, volume 528. John Wiley & Sons, 2005.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

Shiwen Wu, Fei Sun, Wentao Zhang, and Bin Cui. Graph neural networks in recommender systems: a survey. *arXiv preprint arXiv:2011.02260*, 2020a.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020b.

Yu Xie, Yuanqiao Zhang, Maoguo Gong, Zedong Tang, and Chao Han. Mgat: Multi-view graph attention networks. *Neural Networks*, 132:180–189, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Fengyi Zhang, Zhiyong Liu, Fangzhou Xiong, Jianhua Su, and Hong Qiao. Wagnn: A weighted aggregation graph neural network for robot skill learning. *Robotics and Autonomous Systems*, 130:103555, 2020.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020a.

K Zhou, Y Dong, K Wang, WS Lee, B Hooi, H Xu, and J Feng. Understanding and resolving performance degradation in graph convolutional networks. *arXiv preprint arXiv:2006.07107*, 2020b.

Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.