Principal Neighbourhood Aggregation for Graph Nets

Gabriele Corso [*1] **Luca Cavalleri** [*1] **Dominique Beaini** [2] **Pietro Liò** [1] **Petar Veličković** [3]

Abstract

Graph Neural Networks (GNNs) have been shown to be effective models for different predictive tasks on graph-structured data. Recent work on their expressive power has focused on isomorphism tasks and countable feature spaces. We extend this theoretical framework to include continuous features—which occur regularly in real-world input domains and within the hidden layers of GNNs—and we demonstrate the requirement for multiple aggregation functions in this context. Accordingly, we propose Principal Neighbourhood Aggregation (PNA), a novel architecture combining multiple aggregators with degree-scalers (which generalize the sum aggregator). Finally, we compare the capacity of different models to capture and exploit the graph structure via a novel benchmark containing multiple tasks taken from classical graph theory, alongside existing benchmarks from real-world domains, all of which demonstrate the strength of our model.

1. Introduction

Graph Neural Networks (GNNs) have been an active research field for the last ten years with significant advancements in graph representation learning (Scarselli et al., 2009; Bronstein et al., 2017; Hamilton et al., 2017b; Battaglia et al., 2018). However, it is difficult to understand the effectiveness of new GNNs due to the lack of standardized benchmarks (Dwivedi et al., 2020) and of theoretical frameworks for their expressive power.

In fact, most work in this domain has focused on improving the GNN architectures on a set of graph benchmarks. Only recently there have been significant studies on the expressive power of various GNN models (Xu et al., 2018; Morris et al., 2019; Garg et al., 2020). However, these have mainly focused on the isomorphism task in domains with countable features spaces, and little work has been done on understanding their capacity to capture and exploit the underlying properties of the graph structure.

We hypothesize that the aggregation layers of current GNNs are unable to extract enough information from the nodes' neighbourhoods in a single layer, which limits their expressive power and learning abilities.

Then, we mathematically prove the need for multiple aggregators and we propose the concept of degree-scalers, which allow the network to amplify signals. Combining the above, we design the proposed *Principal Neighbourhood Aggregation* (**PNA**).

Finally, we empirically demonstrate the performance of the model on a multi-task benchmark we designed with tasks taken from classical graph theory and on real-world datasets from literature (Dwivedi et al., 2020).

The code for all the aggregators, scalers, models, multi-task and real-world benchmarks is available here.

2. Principal Neighbourhood Aggregation

In this section, we first explain the motivation behind using multiple aggregators concurrently. We then present the idea of degree-based scalers, linking to prior related work on GNN expressiveness. Finally, we detail the design of graph convolutional layers which leverage the proposed Principal Neighbourhood Aggregation.

2.1. Proposed Aggregators

Most work in the literature uses only a single aggregation method, with *mean*, *sum* and *max* aggregators being the most used in the state-of-the-art models (Xu et al., 2018; Kipf & Welling, 2016; Gilmer et al., 2017; Veličković et al., 2019). However, we hypothesize one aggregation function is not enough to discriminate between different neighbourhoods and prove it in the theorem below:

Theorem 1 (Number of aggregators needed)**.** *In order to discriminate between multisets of size n whose underlying set is $\mathbb{R}$, at least n aggregators are needed.*

Proposition 1 (Moments of the multiset)**.** *The moments of a multiset (as defined in Equation 1) exhibit a valid example*

[*]Equal contribution [1]University of Cambridge, Cambridge, United Kingdom [2]Invivo AI, Montreal, Canada [3]DeepMind, London, United Kingdom. Correspondence to: Gabriele Corso <gc579@cam.ac.uk>, Luca Cavalleri <lc737@cam.ac.uk>.

ICML 2020 Workshop on Graph Representation Learning and Beyond. Copyright 2020 by the author(s).

using n aggregators.

We prove Theorem 1 in Appendix A and Proposition 1 in Appendix B. Note that unlike (Xu et al., 2018), we consider a continuous input feature space; this better represents many real-world tasks where the observed values have uncertainty, and better models the latent node features within a neural network's representations. Continuous features make the space uncountable, and void the injectivity proof of the *sum* aggregation presented by (Xu et al., 2018).

Hence, we redefine aggregators as continuous functions of multisets which compute a statistic on the neighbouring nodes, such as *mean*, *max* or *standard deviation*.

Theorem 1 proves that the number of independent aggregators used is a limiting factor of the expressiveness of GNNs. To empirically demonstrate this, we leverage four aggregators, namely *mean*, *maximum*, *minimum* and *std*. In Appendix D we present all these aggregators in detail.

Furthermore, we note that this can be extended to the *normalized moment* aggregators, which allow advanced distribution information to be extracted whenever the degree of the nodes is high. As described in Appendix E, we choose the n$^{\text{th}}$ root normalization, as presented in Equation 1, because it gives a statistic that scales linearly with the size of the individual elements (as the other aggregators); this gives the training adequate numerical stability.

$$M_n(X) = \sqrt[n]{\mathbb{E}\left[(X-\mu)^n\right]} \ , \quad n > 1 \tag{1}$$

2.2. Degree-based Scalers

We introduce scalers as functions of the number of messages being aggregated (usually the node degree), which are multiplied with the aggregated value to perform either an *amplification* or an *attenuation* of the incoming messages.

(Xu et al., 2018) show that the use of *mean* and *max* aggregators by themselves fail to distinguish between neighbourhoods with identical features but with differing cardinalities, and the same applies to all the aggregators described above. They propose the use of the *sum* aggregator to discriminate between such multisets. We generalise their approach by expressing the *sum* aggregator as the composition of a *mean* aggregator and a linear-degree amplifying scaler $S_{\text{amp}}(d) = d$.

Theorem 2 (Injective functions on countable multisets)**.** *The mean aggregation composed with any scaling linear to an injective function on the neighbourhood size can generate injective functions on bounded multisets of countable elements.*

We formalize and prove Theorem 2 in Appendix C.

Recent work (Veličković et al., 2019) shows how *sum* aggregator doesn't generalize to unseen graphs, particularly when larger: a slightly different degree might, in fact, cause the message to vanish or explode. We therefore propose using a logarithmic scaling $S \propto \log(d+1)$ to mitigate this effect. Further motivation for that adoption can be summarized in the following example: consider a social network where nodes A, B and C have respectively 5 million, 1 million and 100 followers: on a linear scale, nodes B and C are the closest pair, while in logarithmic scale, as in human perception, A would be B nearest neighbour.

The logarithmic scaler is generalized in Equation 2, where δ is a normalization parameter computed over the training set, d is the degree of the node receiving the message and α is a variable parameter that is negative for attenuation, positive for amplification or zero for no scaling.

$$S(d,\alpha) = \left(\frac{\log(d+1)}{\delta}\right)^{\alpha}, \ \delta = \frac{1}{|\text{train}|}\sum_{i \,\in\, \text{train}} \log(d_i+1) \tag{2}$$

2.3. Combined Aggregation

We finally introduce the Principal Neighbourhood Aggregation (PNA), a general and flexible architecture, which in our tests we used with four neighbour-aggregations with three degree-scalers each, as summarized in Equation 3, where $\otimes$ is the tensor product:

$$\bigoplus = \underbrace{\begin{bmatrix} I \\ S(D,\alpha=1) \\ S(D,\alpha=-1) \end{bmatrix}}_{\text{scalers}} \otimes \underbrace{\begin{bmatrix} \mu \\ \sigma \\ \max \\ \min \end{bmatrix}}_{\text{aggregators}} \tag{3}$$

We insert the PNA operator within the framework of a message passing neural network (Gilmer et al., 2017), obtaining the following GNN layer:

$$X_i^{(t+1)} = U\left(X_i^{(t)}, \bigoplus_{(j,i)\in E} M\left(X_i^{(t)}, X_j^{(t)}\right)\right) \tag{4}$$

where M and U are neural networks (for our benchmarks, a linear layer was enough). U reduces the size of the concatenated message (in space $\mathbb{R}^{13F}$) back to $\mathbb{R}^F$ where F is the dimension of the hidden features in the network. As in (Gilmer et al., 2017), we employ multiple towers to improve computational complexity and generalization performance.

3. Architecture

We compare the performance of the PNA layer against some of the most popular models in the literature, namely GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017), GIN (Xu et al., 2018) and MPNN (Gilmer et al., 2017) on a common architecture. In Appendix F, we present the details of these graph convolutional layers.

For the multi-task experiments, we used an architecture, represented in Figure 1, with $\mathcal{M}$ convolutions followed by three fully-connected layers for node labels and a set2set (S2S) (Vinyals et al., 2015) readout function for graph labels. In particular, we want to highlight:

Gated Recurrent Units (GRU) (Cho et al., 2014) applied after the update function of each layer, as in (Gilmer et al., 2017; Li et al., 2015). Their ability to retain information from previous layers proved effective when increasing the number of convolutional layers $\mathcal{M}$.

Weight Sharing in all the GNN layers but the first makes the architecture follow an encode-process-decode configuration (Battaglia et al., 2018; Hamrick et al., 2018). This is a strong prior which works well on all our experimental tasks, yields a parameter-efficient architecture, and allows the model to have a variable number $\mathcal{M}$ of layers.

Variable Depth $\mathcal{M}$, decided at inference time (based on the size of the input graph and/or other heuristics), is important when using models over high variance graph distributions. In our experiments we have only used heuristics dependant on the number of nodes N ($\mathcal{M} = f(N)$) and, for the architectures in the results below, we settled with $\mathcal{M} = \lfloor N/2 \rfloor$.

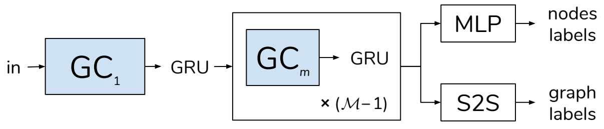


Figure 1. Layout of the architecture used. When comparing different models, the difference lies only in the type of graph convolution used in place of GC_1 and GC_m.

4. Benchmarks and Results

4.1. Multi-tasks Artificial Benchmark

We developed a multi-task benchmark with tasks from classical graph theory to test the model understanding of graph features. In particular, we generated random graphs from a wide variety of types and trained the models on node labels representing single-source shortest-path lengths, the eccentricity and the Laplacian features and graph labels representing whether the graph is connected, the diameter and the spectral radius. Further details on the tasks, graphs generation, features and training settings can be found in Appendix G.

The results are presented in Figures 2 and 3, where we observe that the proposed PNA model consistently outperforms state-of-the-art models. The *baseline* predicts the average of the training set for all tasks.

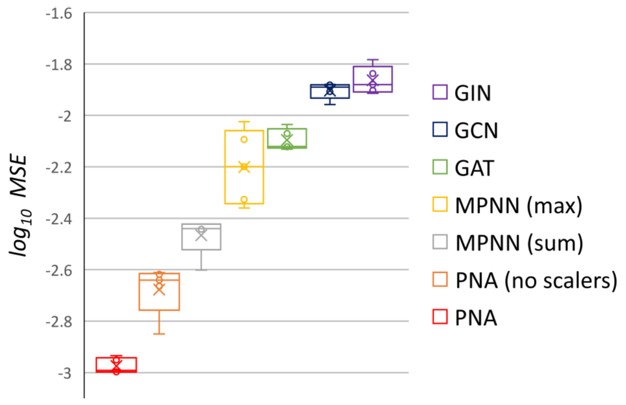
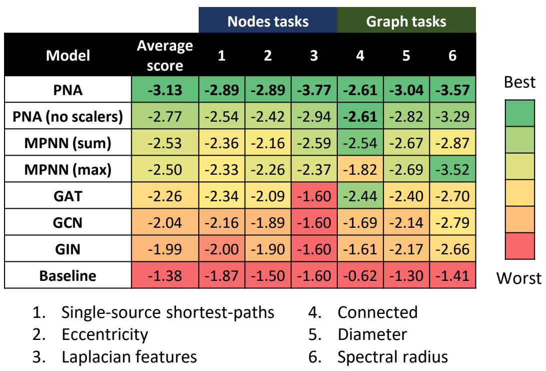


Figure 2. Distribution of the $\log_{10}$MSE errors for the top 5 performances of each model on the multi-task benchmark for different GNN models using the same architecture and various near-optimal hyper-parameters.



		Nodes tasks			Graph tasks		
Model	**Average score**	**1**	**2**	**3**	**4**	**5**	**6**
PNA	**-3.13**	**-2.89**	**-2.89**	**-3.77**	**-2.61**	**-3.04**	**-3.57**
PNA (no scalers)	-2.77	-2.54	-2.42	-2.94	**-2.61**	-2.82	-3.29
MPNN (sum)	-2.53	-2.36	-2.16	-2.59	-2.54	-2.67	-2.87
MPNN (max)	-2.50	-2.33	-2.26	-2.37	-1.82	-2.69	-3.52
GAT	-2.26	-2.34	-2.09	-1.60	-2.44	-2.40	-2.70
GCN	-2.04	-2.16	-1.89	-1.60	-1.69	-2.14	-2.79
GIN	-1.99	-2.00	-1.90	-1.60	-1.61	-2.17	-2.66
Baseline	-1.38	-1.87	-1.50	-1.60	-0.62	-1.30	-1.41

Best
Worst

1. Single-source shortest-paths
2. Eccentricity
3. Laplacian features
4. Connected
5. Diameter
6. Spectral radius

Figure 3. Mean $\log_{10}$MSE error for each task and their average.

The multi-task results follow and amplify the PNA superiority arising in single-task training, suggesting deeper exploitation of common sub-units, which the other models cannot achieve even when their latent size is significantly increased, as shown by further experiments presented in Appendix H.

Finally, we explored the extrapolation of the models to larger graphs, (15-25 training, 25-30 validation and 20-50 test). Unlike in (Veličković et al., 2019), the models are not given any step-wise supervision or trained on easily extendable subroutines, and they have to cope with their architectures being augmented with further hidden layers. The results in Figure 4 show that the PNA outperforms other models and avoids the explosion issue encountered by some models using the *sum* aggregator.

Model		ZINC		CIFAR10		MNIST	
		No edge features	Edge features	No edge features	Edge features	No edge features	Edge features
		MAE	MAE	Acc	Acc	Acc	Acc
Dwivedi et al. paper	MLP	0.710±0.001		56.01±0.90		94.46±0.28	
	MLP (Gated)	0.681±0.005		56.78±0.12		95.18±0.18	
	GCN	0.469±0.002		54.46±0.10		89.99±0.15	
	GraphSage	0.410±0.005		66.08±0.24		97.20±0.17	
	GIN	0.408±0.008		53.28±3.70		93.96±1.30	
	DiffPoll	0.466±0.006		57.99±0.45		95.02±0.42	
	GAT	0.463±0.002		65.48±0.33		95.62±0.13	
	MoNet	0.407±0.007		53.42±0.43		90.36±0.47	
	GatedGCN	0.422±0.006	0.363±0.009	69.19±0.28	69.37±0.48	97.37±0.06	97.47±0.13
Our experi-ments	MPNN (sum)	0.381±0.005	0.288±0.002*	65.39±0.47	65.61±0.30	96.72±0.17	96.90±0.15
	MPNN (max)	0.468±0.002	0.328±0.008*	69.70±0.55	**70.86±0.27**	97.37±0.11	97.82±0.08
	PNA (no scalers)	0.413±0.006	0.247±0.036*	**70.46±0.44**	70.47±0.72	**97.41±0.16**	**97.94±0.12**
	PNA	**0.320±0.032**	**0.188±0.004***	70.21±0.15	70.35±0.63	97.19±0.08	97.69±0.22

Figure 5. Results of the PNA and MPNN models in comparison with those analysed by Dwivedi *et al.* (Kipf & Welling, 2016; Hamilton et al., 2017a; Xu et al., 2018; Ying et al., 2018; Veličković et al., 2017; Monti et al., 2017; Bresson & Laurent, 2017). * indicates the training was conducted with additional patience to ensure convergence.

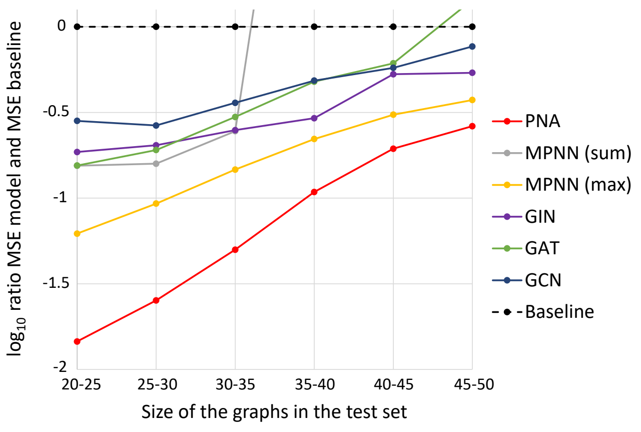


Figure 4. Multi-task $\log_{10}$ of the ratio of the MSE for different GNN models and the MSE of the baseline.

4.2. Real-world Benchmarks

To test the capacity of the PNA model in real-world domains, we assess it on the chemistry (ZINC) and computer vision (CIFAR10 and MNIST) benchmark proposed in (Dwivedi et al., 2020). To ensure a fair comparison, we followed their method for training procedure and GNN structure.

To better understand the results in Figure 5, we need to take into account how graphs differ among the three datasets. In the chemistry benchmark, graphs are diverse and individual edges (bonds) can significantly impact the properties of the graphs (molecules). This contrasts with computer vision datasets made of graphs with a regular topology (every node has 8 edges) and where the graph structure of the representation is not crucial (the good performance of the MLP is evidence).

With this and our theoretical analysis in mind, it is understandable why the PNA has a strong performance in the chemistry datasets, as it was designed to understand the graph structure and better retain neighbourhood information. At the same time, the version without scalers suffers from the fact it cannot distinguish between neighbourhoods of different size. Instead, in the computer vision datasets, the average improvement of the PNA on SOTA was lower due to the smaller importance of the graph structure, and the version of the PNA without scalers performs better as the constant degree of these graphs makes scalers redundant (and it is preferable to 'spend' parameters for larger hidden sizes).

5. Conclusion

We have extended the theoretical framework in which GNNs are analyzed to continuous features proving the need for multiple aggregators in such circumstances, and we presented a method, Principal Neighbourhood Aggregation, consisting of the composition of multiple aggregators and degree-scalers. With the goal of understanding the ability of GNNs to capture graph structures, we have proposed a novel multi-task benchmark and an encode-process-decode architecture for approaching it. Empirical results from synthetic and real-world domains support our theoretical evidence. We believe that our findings constitute a step towards establishing a hierarchy of models w.r.t. their expressive power, where the PNA model appears to outperform the prior art in GNN layer design.

References

Albert, R. and Barabási, A.-L. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74 (1):47–97, Jan 2002. ISSN 1539-0756. doi: 10.1103/revmodphys.74.47. URL http://dx.doi.org/10.1103/RevModPhys.74.47.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Borsuk, K. Drei sätze über die n-dimensionale euklidische sphäre. *Fundamenta Mathematicae*, (20):177–190, 1933. doi: 10.4064/fm-20-1-177-190.

Bresson, X. and Laurent, T. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

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, Jul 2017. ISSN 1053-5888. doi: 10.1109/msp.2017.2693418. URL http://dx.doi.org/10.1109/MSP.2017.2693418.

Chen, Z., Chen, L., Villar, S., and Bruna, J. Can graph neural networks count substructures? *arXiv preprint arXiv:2002.04025*, 2020.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Erdős, P. and Rényi, A. On the evolution of random graphs. pp. 17–61, 1960.

Garg, V. K., Jegelka, S., and Jaakkola, T. Generalization and representational limits of graph neural networks. *arXiv preprint arXiv:2002.06157*, 2020.

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

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

Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017b.

Hamrick, J. B., Allen, K. R., Bapst, V., Zhu, T., McKee, K. R., Tenenbaum, J. B., and Battaglia, P. W. Relational inductive bias for physical construction in humans and machines. *arXiv preprint arXiv:1806.01203*, 2018.

Hazewinkel, M. *Encyclopaedia of mathematics. Volume 9, STO-ZYG.* Encyclopaedia of mathematics ; vol 9: STO-ZYG. Kluwer Academic, Dordecht, 1988. ISBN 1556080085.

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

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

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.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

Rotman, J. J. *An Introduction to Algebraic Topology*, volume 119 of *Graduate Texts in Mathematics*. Springer New York. ISBN 978-1-4612-8930-2 978-1-4612-4576-6. doi: 10.1007/978-1-4612-4576-6. URL http://link.springer.com/10.1007/978-1-4612-4576-6.

Sato, R., Yamada, M., and Kashima, H. Random features strengthen graph neural networks. *arXiv preprint arXiv:2002.03155*, 2020.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Stanley, R. P. *Enumerative Combinatorics Volume 2*. Cambridge Studies in Advanced Mathematics no. 62. Cambridge University Press, Cambridge, 2001. ISBN 9780511609589.

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

Veličković, P., Ying, R., Padovano, M., Hadsell, R., and Blundell, C. Neural execution of graph algorithms. *arXiv preprint arXiv:1910.10593*, 2019.

Vinyals, O., Bengio, S., and Kudlur, M. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

Watts, D. J. Networks, dynamics, and the small-world phenomenon. *American Journal of Sociology*, 105(2): 493–527, 1999. ISSN 00029602, 15375390. URL `http://www.jstor.org/stable/10.1086/210318`.

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

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.

You, J., Ying, R., and Leskovec, J. Position-aware graph neural networks. *arXiv preprint arXiv:1906.04817*, 2019.

A. Proof for Theorem 1 (Number of aggregators needed)

In order to discriminate between multisets of size n whose underlying set is $\mathbb{R}$, at least n aggregators are needed.

Proof. Let S be the n-dimensional subspace of $\mathbb{R}^n$ formed by all tuples $(x_1, x_2, \ldots, x_n)$ such that $x_1 \leq x_2 \leq \ldots \leq x_n$, and notice how S is the collection of the aforementioned multisets. We defined an aggregator as a continuous function from multisets to reals, which corresponds to a continuous function $g : S \to \mathbb{R}$.

Assume by contradiction that it is possible to discriminate between all the multisets of size n using only $n-1$ aggregators, viz. $g_1, g_2, \ldots, g_{n-1}$.

Define $f{:}S{\to}\mathbb{R}^{n-1}$ to be the function mapping each multiset X to its output vector $(g_1(X),\, g_2(X),\, \ldots\,, g_{n-1}(X))$. Since $g_1, g_2, \ldots, g_{n-1}$ are continuous, so is f, and, since we assumed these aggregators are able to discriminate between all the multisets, f is injective.

As S is a n-dimensional Euclidean subspace, it is possible to define a $(n-1)$-sphere C^{n-1} entirely contained within it, i.e. $C^{n-1} \subseteq S$. According to Borsuk–Ulam theorem (Rotman; Borsuk, 1933), there are two distinct (in particular, non-zero and antipodal) points $\vec{x}_1, \vec{x}_2 \in C^{n-1}$ satisfying $f(\vec{x}_1) = f(\vec{x}_2)$, showing f not to be injective; hence the required contradiction. $\square$

Note: n aggregators are actually sufficient. A simple example is to use $g_1, g_2, \ldots, g_n$ where $g_k(X) =$ the k-th smallest item in X. It's clear to see that the multiset whose elements are $g_1(X),\, g_2(X),\, \ldots\,, g_n(X)$ is X, which can hence be uniquely determined by the aggregators.

B. Proof for Proposition 1 (Moments of the multiset)

The moments of a multiset (as defined in Equation 1) exhibit a valid example using n aggregators.

Proof. Since $n \geq 1$, and the first aggregator is *mean*, we know μ. Let $X = \{x_1, x_2, \ldots, x_n\}$ be the multiset to be found, and define $R = \{r_1 = x_1 - \mu,\ r_2 = x_2 - \mu,\ \ldots,\ r_n = x_n - \mu\}$.

Notice how $\sum {r_i}^1 = 0$, and for $1 < k \leq n$ we have $\sum {r_i}^k = n\ M_k(X)^k$, i.e. all the symmetric power sums $p_k = \sum {r_i}^k$ $(k \leq n)$ are uniquely determined by the moments.

Additionally, e_k, the elementary symmetric sums of R, i.e. the sum of the products of all the sub-multisets of size k $(1 \leq k \leq n)$, are determined as follow:

e_1, the sum of all elements, is equal to p_1; e_2, the sum of the products of all pairs in R, is $(e_1 p_1 - p_2)/2$; e_3, the sum of the products of all triplets, is $(e_2 p_1 - e_1 p_2 + p_3)/3$, and so on:

$$e_k = \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq n} \left(\prod_{j=1}^{k} r_{i_j} \right) \quad , \quad e_0 = 1$$

Notice how $e_1, e_2, \ldots, e_n$ can be computed using the following recursive formula (Stanley, 2001):

$$e_k = \frac{1}{k} \sum_{j=1}^{k} (-1)^{j-1} e_{k-j} p_j$$

Consider polynomial $P(x) = \Pi(x - r_i)$, i.e. the unique polynomial of degree n with leading coefficient 1 whose roots are R. This defines A, the coefficients of P, i.e. the real numbers $a_0, a_1, \ldots, a_{n-1}$ for which $P(x) = x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$. Using Vieta's formulas (Hazewinkel, 1988):

$$\sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq n} \left(\prod_{j=1}^{k} r_{i_j} \right) = (-1)^k \frac{a_{n-k}}{a_n}$$

we obtain

$$\begin{aligned} e_k &= (-1)^k \frac{a_{n-k}}{a_n} \\ &= (-1)^k a_{n-k} && \text{recall } a_n = 1 \\ \therefore\ a_i &= (-1)^{n+i} e_{n+i} && \text{letting } k = n + i \text{ and rearranging} \end{aligned}$$

Hence A is uniquely determined, and so is P, being its coefficients a valid definition of it. By the fundamental theorem of algebra, P has n (possibly repeated) roots, which are the elements of R, hence uniquely determining the latter.

Finally, X can be easily determined adding μ to each element of R. $\square$

Note: the proof above assumes the knowledge of n. In the case that n is variable (as in GNNs), and so we have multisets of up to n elements, an extra aggregator will be needed. An example of such aggregator is the *mean* multiplied by any injective scaler which would allow the degree of the node to be inferred.

C. Proof for Theorem 2 (Injective functions on countable multisets)

The mean aggregation composed with any scaling linear to an injective function on the neighbourhood size can generate injective functions on bounded multisets of countable elements.

Proof. Let χ be the countable input feature space from which the elements of the multisets are taken and X an arbitrary multiset. Since χ is countable and the cardinality of multisets is bounded, let $Z : \chi \rightarrow \mathbb{N}^{+}$ be an injection from χ to natural numbers, and $N \in \mathbb{N}$ such that $|X| + 1 < N$ for all X.

Let's define an injective function s, and without loss of generality, assume $s(0), s(1), \ldots, s(N) > 0$ (otherwise for the rest of the proof consider s as $s'(i) = s(i) - \min_{j\in[0,N]} s(j) + \epsilon$ which is positive for all $i \in [0, N]$). $s(|X|)$ can only take value in $\{s(0), s(1), \ldots, s(N)\}$, therefore let us define $\gamma = \min\left\{\frac{s(i)}{s(j)} \mid i, j \in [0, N],\ s(i) \geq s(j)\right\}$. Since s is injective, $s(i) \neq s(j)$ for $i \neq j$, which implies $\gamma > 1$.

Let $K > \frac{1}{\gamma - 1}$ be a positive real number and consider $f(x) = N^{-Z(x)} + K$.

$\forall x \in \chi, Z(x) \in [1, N] \Rightarrow N^{-Z(x)} \in [0, 1] \Rightarrow f(x) \in [K, K+1]$, so $\mathbb{E}_{x\in X}[f(x)] \in [K, K+1]$.

We proceed to show that the cardinality of X can be uniquely determined, and X itself can be determined as well, by showing that exist an injection h over the multisets.

Let us h as a function that scales the mean of f by an injective function of the cardinality:

$$h(X) = s(|X|)\, \mathbb{E}_{x\in X}[f(x)]$$

We want show that the value of $|X|$ can be uniquely inferred from the value of $h(X)$. Assume by contradiction $\exists\, X', X''$ multisets of size at most N such that $|X'| \neq |X''|$ but $h(X') = h(X'')$; since s is injective $s(|X'|) \neq s(|X''|)$, without loss of generality let $s(|X'|) > s(|X''|)$, then:

$$\begin{aligned} s(|X''|)(K+1) \geq s(|X''|)\, \mathbb{E}_{x\in X''}[f(x)] = h(X'') = \\ = h(X') = s(|X'|)\, \mathbb{E}_{x\in X'}[f(x)] \geq s(|X'|)\, K \\ \implies K \leq \frac{1}{\frac{s(|X'|)}{s(|X''|)} - 1} \leq \frac{1}{\gamma - 1} \end{aligned}$$

which is a contradiction. So it is impossible for the size of a multiset X to be ambiguous from the value of $h(X)$.

Let us define d as the function mapping $h(X)$ to $|X|$.

$$\begin{aligned} h'(X) = \sum_{x\in X} N^{-Z(x)} = \frac{h(X)|X|}{s(|X|)} - K|X| = \\ = \frac{h(X)d(h(X))}{s(d(h(X)))} - Kd(h(X)) \end{aligned}$$

Considering the $Z(j)$-th digit i after the decimal point in the base N representation of $h'(X)$, it can be inferred that X contains i elements j, and, so, all the elements in X can be determined; hence h is injective over the multisets in X. $\square$

Note: this proof is a generalization of the one by *Xu et al.* (Xu et al., 2018) on the *sum* aggregator.

D. Aggregators

The following paragraphs will describe in detail the aggregators we leveraged in our architectures.

Mean Aggregation $\mu(X^l)$ The most common message aggregator in the literature, wherein each node computes a weighted average or sum of its incoming messages. Equation 5 presents, on the left, the general mean equation, and, on the right, the direct neighbour formulation, where X is any multiset, X^l are the nodes' features at layer l, $N(i)$ is the neighbourhood of node i and $d_i = |N(i)|$. For clarity we use $\mathbb{E}[f(X)]$ where X is a multiset of size d to be defined as $\mathbb{E}[f(X)] = \frac{1}{d}\sum_{x\in X} f(x)$.

$$\mu(X) = \mathbb{E}[X] \quad , \quad \mu_i(X^l) = \frac{1}{d_i}\sum_{j\in N(i)} X_j^l \tag{5}$$

Maximum and Minimum Aggregations $\max(X^l)$, $\min(X^l)$ Also often used in literature, they are very useful for discrete tasks, for domains where credit assignment is important and when extrapolating to unseen distributions of graphs (Veličković et al., 2019). Alternatively, we present the softmax and softmin aggregators in Appendix D, which are differentiable and work for weighted graphs, but don't perform as well on our benchmarks.

$$\max_i(X^l) = \max_{j\in N(i)} X_j^l\ , \quad \min_i(X^l) = \min_{j\in N(i)} X_j^l \tag{6}$$

Standard Deviation Aggregation $\sigma(X^l)$ The standard deviation (STD or σ) is used to quantify the spread of neighbouring nodes features, such that a node can assess the diversity of the signals it receives. Equation 7 presents first the standard deviation formulation, and then the STD of a graph-neighbourhood. *ReLU* is the rectified linear unit used to avoid negative values caused by numerical errors and ϵ is a small positive number to ensure σ is differentiable.

$$\begin{aligned} \sigma(X) = \sqrt{\mathbb{E}[X^2] - \mathbb{E}[X]^2} \quad , \\ \sigma_i(X^l) = \sqrt{ReLU\left(\mu_i({X^l}^2) - \mu_i(X^l)^2\right) + \epsilon} \end{aligned} \tag{7}$$

Normalized Moments Aggregation $M_n(X^l)$ The mean and standard deviation are the first and second normalized moments of the multiset ($n = 1, n = 2$). Additional moments, such as the skewness ($n = 3$), the kurtosis ($n = 4$), or higher moments, could be useful to better describe the neighbourhood. These become even more important when the degree of a node is high because four aggregators are insufficient to describe the neighbourhood accurately. As described in Appendix E, we choose the nth root normalization, as presented in Equation 8, because it gives a statistic that scales linearly with the size of the individual elements (as the other aggregators); this gives the training adequate numerical stability. Once again we add an ϵ to the absolute value of the expectation before applying the nth root for numerical stability of the gradient.

$$M_n(X) = \sqrt[n]{\mathbb{E}\left[(X-\mu)^n\right]} \quad , \quad n > 1 \tag{8}$$

Besides those described above, we have experimented with additional aggregators. We detail some examples below. Domain-specific metrics can also be an effective choice.

Softmax and Softmin Aggregations As an alternative to *max* and *min*, *softmax* and *softmin* are differentiable and can be weighted in the case of edge features or attention networks. They also allow an asymmetric message passing in the direction of the strongest signal.

$$\begin{aligned} \text{softmax}_i(X^l) &= \sum_{j \in N(i)} \frac{X_j^l \exp(X_j^l)}{\sum_{k \in N(i)} \exp(X_k^l)} \quad , \\ \text{softmin}_i(X^l) &= -\text{softmax}_i(-X^l) \end{aligned} \tag{9}$$

E. Normalized Moments Aggregation

The main motivation for choosing the nth root normalization for the moments is numerical stability. In fact, one property of our version is that it scales linearly with L, for uniformly distributed random variables $U[0, L]$, as do other aggregators such as mean, max and min (std is a particular case). Other common formulations of the moments such as those in Equation 10 scale respectively as the nth power and constantly with L. This difference causes numerical instability when combined in the same layer.

$$M_n(X) = \mathbb{E}\left[(X-\mu)^n\right] \qquad M_n(X) = \frac{\mathbb{E}\left[(X-\mu)^n\right]}{\sigma^n} \tag{10}$$

To demonstrate the usefulness of higher moments aggregation and further motivate the need for multiple aggregation functions, we ran an ablation study showing how different moments affect the performance of the model. We conduct this by testing five different models, each taking a different number of moments, on our multi-task benchmark.

The results in Figure 6 demonstrate that with the increase of the number of aggregators the models reach a higher expressive power, but at a certain point (dependent on the graphs and tasks, in this case around 3) the increase in expressiveness given by higher moments reduces the performance

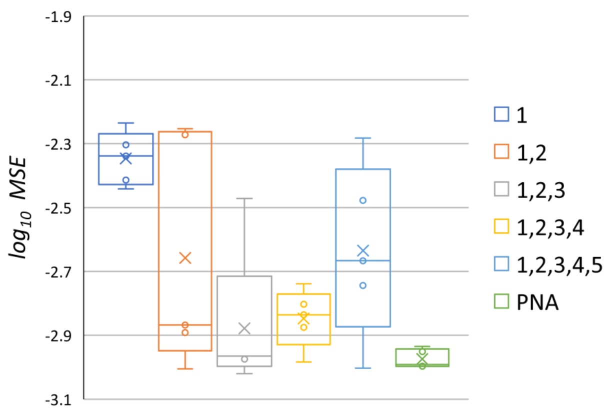


Figure 6. Multi-task $\log_{10}$ MSE on different versions of the PNA model with increasing number of moments aggregators (specified in the legend), using *mean* as first moment. All the models use the identity, amplification and attenuation scalers. The model on the right is the complete PNA as described before (*mean*, *max*, *min* and *std* aggregators).

since the model becomes harder to optimize and prone to overfitting. We expect that higher moments will be more beneficial on graphs with a higher average degree since they will better characterize the neighbourhood distributions.

Finally, we note how the addition of the *max* and *min* aggregators in the PNA (rightmost column) gives a better and more consistent performance in these tasks than higher moments. We believe this is task-dependent, and, for algorithmic tasks, discrete aggregators can be valuable. As a side note, we point out how the *max* and *min* aggregators of positive values can be considered as the nth-root of the nth (non-centralized) moment as n tends to, respectively, $+\infty$ and $-\infty$.

F. Alternative Graph Convolutions

In this section, we present the details of the four graph convolutional layers from existing models that we used to compare the performance of the PNA in the multi-task benchmark.

Graph Convolutional Networks (GCN) (Kipf & Welling, 2016) use a normalized mean aggregator followed by a linear transformation and an activation function. We define it in Equation 11, where $\tilde{A} = A + I_N$ is the adjacency matrix with self-connections, W is a trainable weight matrix and b a learnable bias.

$$X^{(t+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X^{(t)} W + b\right) \tag{11}$$

Graph Attention Networks (GAT) (Veličković et al., 2017) perform a linear transformation of the input features followed by an aggregation of the neighbourhood as a weighted sum of the transformed features, where the weights are set by an attention mechanism a. We define it in Equation 12, where W is a trainable projection matrix. As in the original paper, we employ the use of multi-head attention.

$$X_i^{(t+1)} = \sigma\left(\sum_{(j,i)\in E} a\left(X_i^{(t)}, X_j^{(t)}\right) W X_j^{(t)}\right) \tag{12}$$

Graph Isomorphism Networks (GIN) (Xu et al., 2018) perform a sum aggregation over the neighbourhood, followed by an update function U consisting of a multi-layer perceptron. We define it in Equation 13, where ϵ is a learnable parameter. As in the original paper, we use a 2-layer MLP for U.

$$X_i^{(t+1)} = U\left(\left(1+\epsilon\right) X_i^{(t)} + \sum_{j\in N(i)} X_j^{(t)}\right) \tag{13}$$

Message Passing Neural Networks (MPNN) (Gilmer et al., 2017) perform a transformation before and after an arbitrary aggregator. We define it in Equation 14, where M and U are neural networks and $\bigoplus$ is a single aggregator. In particular, we test models with *sum* and *max* aggregators, as they are the most used in literature. As with PNA layers, we found that linear transformations are sufficient for M and U and, as in the original paper (Gilmer et al., 2017), we employ multiple towers.

$$X_i^{(t+1)} = U\left(X_i^{(t)}, \bigoplus_{(j,i)\in E} M\left(X_i^{(t)}, X_j^{(t)}\right)\right) \tag{14}$$

G. Multi-task Benchmark

The benchmark consists of classical graph theory tasks on artificially generated graphs.

Random Graph Generation Following previous work (Veličković et al., 2019; You et al., 2019), the benchmark contains undirected unweighted randomly generated graphs of a wide variety of types (we provide, in parentheses, the approximate proportion of such graphs in the benchmark). Letting N be the total number of nodes per graph:

- **Erdős-Rényi** (Erdős & Rényi, 1960) (20%): with probability of presence for each edge equal to p, where p is independently generated for each graph from $\mathcal{U}[0,1]$
- **Barabási-Albert** (Albert & Barabási, 2002) (20%): the number of edges for a new node is k, which is taken randomly from $\{1, 2, ..., N-1\}$ for each graph
- **Grid** (5%): $m \times k$ 2d grid graph with $N = mk$ and m and k as close as possible
- **Caveman** (Watts, 1999) (5%): with m cliques of size k, with m and k as close as possible
- **Tree** (15%): generated with a power-law degree distribution with exponent 3
- **Ladder graphs** (5%)
- **Line graphs** (5%)
- **Star graphs** (5%)
- **Caterpillar graphs** (10%): with a backbone of size b (drawn from $\mathcal{U}[1, N)$), and $N-b$ pendent vertices uniformly connected to the backbone
- **Lobster graphs** (10%): with a backbone of size b (drawn from $\mathcal{U}[1, N)$), p (drawn from $\mathcal{U}[1, N-b]$) pendent vertices uniformly connected to the backbone, and additional $N-b-p$ pendent vertices uniformly connected to the previous pendent vertices.

Additional randomness was introduced to the generated graphs by randomly toggling arcs, without strongly impacting the average degree and main structure. If e is the number of edges and m the number of 'missing edges' ($2e + 2m = N(N-1)$), the probabilities P_e and P_m of an existing and missing edge to be toggled are:

$$P_e = \begin{cases} 0.1 & e \le m \\ 0.1\,\frac{m}{e} & e > m \end{cases} \qquad P_m = \begin{cases} 0.1\,\frac{e}{m} & e \le m \\ 0.1 & e > m \end{cases} \tag{15}$$

After performing the random toggling, we discarded graphs containing singleton nodes, as they are in no way affected by the choice of aggregation.

For the presented multi-task results, we used graphs of small sizes (15 to 50 nodes) as they were already sufficient to demonstrate clear differences between the models.

Graph Properties In the multi-task benchmark, we consider three node labels and three graph labels based on standard graph theory problems. The node properties tasks are the single-source shortest-path lengths, the eccentricity and the Laplacian features (LX where $L = (D - A)$ is the Laplacian matrix and X the node feature vector). The graph properties tasks are whether the graph is connected, the diameter and the spectral radius.

Input Features As input features, the network is provided with two vectors of size N, a one-hot vector (representing the source for the shortest-path task) and a feature vector X where each element is i.i.d. sampled as $X_i \sim \mathcal{U}[0,1]$. Apart from taking part in the Laplacian features task, this random feature vector also provides a "unique identifier" for the nodes in other tasks. Similar strengthening via random features was also concurrently discovered by (Sato et al., 2020). This allows for addressing some of the problems highlighted in (Garg et al., 2020; Chen et al., 2020); e.g. the task of whether a graph is connected could be performed by continually aggregating the maximum feature of the neighbourhood and then checking whether they are all equal in the readout.

Model Training While having clear differences, these tasks also share related subroutines (such as graph traversals). While we do not take this sharing of subroutines as prior as in (Veličković et al., 2019), we expect models to pick up on these commonalities and efficiently share parameters between the tasks, which reinforce each other during the training.

We trained the models using the Adam optimizer for a maximum of 10,000 epochs, using early stopping with a patience of 1,000 epochs. Learning rates, weight decay, dropout and other hyper-parameters were tuned on the validation set. For each model, we run 10 training runs with different seeds and different hyper-parameters (but close to the tuned values) and report the five with least validation error.

H. Parameters Comparison

Table 1 shows the results of testing all the other models on the multi-task benchmark with increased latent size.

Table 1. Average score of different models using feature sizes of 16 and 20, compared to the PNA with 16 on the multi-task benchmark. "# params" is the total number of parameters in each architecture.

	SIZE 16		SIZE 20	
MODEL	# PAR	log MSE	# PAR	log MSE
PNA	8350	**-3.13**	-	-
MPNN (SUM)	7294	-2.53	11186	-2.19
MPNN (MAX)	8032	-2.50	12356	**-2.23**
GAT	6694	-2.26	10286	-2.08
GCN	6662	-2.04	10246	-1.96
GIN	7272	-1.99	11168	-1.91

We observe that, even with fewer parameters, PNA performs consistently better and an increased number of parameters does not boost the performance of the other models. This suggests that the multiple aggregators in the PNA produce a qualitative improvement to the capacity of the model.