GNN-VPA: A VARIANCE-PRESERVING AGGREGATION STRATEGY FOR GRAPH NEURAL NETWORKS

Lisa Schneckenreiter[1]*, **Richard Freinschlag**[1]*, **Florian Sestak**[1],
Johannes Brandstetter[1,2], **Günter Klambauer**[1], **Andreas Mayr**[1]
[1] ELLIS Unit Linz and LIT AI Lab, Institute for Machine Learning,
Johannes Kepler University, Linz, Austria
[2] NXAI GmbH, Linz, Austria
{last-name}@ml.jku.at

ABSTRACT

The successful graph neural networks (GNNs) and particularly message passing neural networks critically depend on the functions employed for message aggregation and graph-level readout. Using signal propagation theory, we propose a variance-preserving aggregation function, which maintains the expressivity of GNNs while improving learning dynamics. Our results could pave the way towards normalizer-free or self-normalizing GNNs.

1 INTRODUCTION

For many real-world prediction tasks, graphs naturally represent the input data. Graph neural networks (GNNs) (Scarselli et al., 2009; Kipf & Welling, 2017; Defferrard et al., 2016; Veličković et al., 2018) are therefore of large interest as they are able to naturally process such data.

While the expressivity of GNNs has been investigated profoundly (Xu et al., 2019), signal propagation (Neal, 1995; Schoenholz et al., 2017; Klambauer et al., 2017) through GNNs is currently underexplored. With simplistic assumptions, we motivate the use of a variance-preserving aggregation function for GNNs (see Fig. 1), which maintains expressivity while improving signal propagation and consequently learning dynamics (see Apps. A.1 and A.2 for further background details).

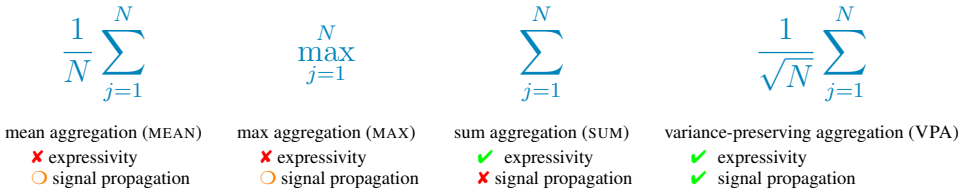

Figure 1: Overview of main message aggregation functions and their properties.

2 GNNS WITH VARIANCE PRESERVATION

Graph neural networks (GNNs) exchange information, i.e., messages, between neighboring nodes of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $v_i \in \mathcal{V}$ and edges $e_{ij} \in \mathcal{E}$ through the application of a local, permutation-invariant function to all neighborhoods. The core layers iteratively update node embeddings $\boldsymbol{h}_i \in \mathbb{R}^D$ at node v_i via three substeps 1.-3.:

$$1.\ \boldsymbol{m}_{ij} = \phi\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) \ \text{ or } \ \boldsymbol{m}_{ij} = \phi\left(\boldsymbol{h}_j\right) \quad 2.\ \boldsymbol{m}_i^{\oplus} = \oplus_{j \in \mathcal{N}(i)}\, \boldsymbol{m}_{ij} \quad 3.\ \boldsymbol{h}_i' = \psi\left(\boldsymbol{h}_i, \theta\left(\boldsymbol{m}_i^{\oplus}\right)\right)$$

to a new embedding $\boldsymbol{h}_i'$, where the aggregation $\oplus_{j \in \mathcal{N}(i)}$ at node v_i is across all neighboring nodes $\mathcal{N}(i)$, i.e., those nodes v_j, that are connected to node v_i via an edge e_{ij}. These nodes are renumbered

*Equal contribution

according to Fig. 1 from 1 to N, with $N := |\mathcal{N}(i)|$. Depending on the type of GNN, ϕ, ψ, and θ can be realized as learnable functions, usually Multilayer Perceptrons (MLPs). E.g., for Graph Convolutional Networks (GCNs) (Kipf & Welling, 2017) only ψ is learnable, for general Message Passing Neural Networks (MPNNs) (Gilmer et al., 2017) ϕ and ψ are learnable, and for Graph Isomorphism Networks (GINs) (Xu et al., 2019) all three are learnable.

Signal propagation theory allows to analyze the distribution of quantities through randomly initialized neural networks. From certain assumptions (for details see App. B.1) it follows that $\boldsymbol{m}_{ij} \sim p_{\mathcal{N}}(\mathbf{0}, \boldsymbol{I})$. If $\boldsymbol{m}_{ij}$ are further assumed to be independent of each other[1], one obtains $\boldsymbol{m}_i^{\text{SUM}} \sim p_{\mathcal{N}}(\mathbf{0}, N\boldsymbol{I})$ for SUM aggregation (i.e., $\oplus \equiv \sum_{i=1}^{N}$), and $\boldsymbol{m}_i^{\text{MEAN}} \sim p_{\mathcal{N}}(\mathbf{0}, \frac{1}{N}\boldsymbol{I})$ for MEAN aggregation (i.e., $\oplus \equiv \frac{1}{N}\sum_{i=1}^{N}$) at initialization.

Our key idea is to introduce a new aggregation function which preserves variance, i.e., $\boldsymbol{m}_i \sim p_{\mathcal{N}}(\mathbf{0}, \boldsymbol{I})$. This is possible with $\oplus \equiv \frac{1}{\sqrt{N}}\sum_{i=1}^{N}$. We denote this function as *variance-preserving aggregation* (VPA) and show the preservation property by applying Lemma B1 element-wise.

Note that by using VPA as aggregation function, the variance of messages theoretically remains in the same range throughout the message passing layers in contrast to SUM or MEAN. According to signal propagation theory such behavior is advantageous for learning. We further note, that for a message passing algorithm with VPA, the expressive power w.r.t. discriminating non-isomorphic graphs does not decrease compared to using SUM (see Lemma B2 in App. B.3) and is stronger than with MEAN.

3 Experiments

We tested the effectiveness of our idea with established GNN architectures on the same graph classification benchmarks as Xu et al. (2019). To evaluate prediction performance, we combined the GIN and the GCN architectures with each of the aggregation methods in Fig. 1 both for message aggregation and graph-level read-out and report the test accurracies of a 10-fold cross-validation. Further, we compared the original Simple Graph Convolution (SGC) (Wu et al., 2019) and Graph Attention Networks (GAT) (Veličković et al., 2018) with modified versions incorporating a tailored variance-preserving aggregation approach (see App. C.1 and App. B.4 for details). The results are shown in Table 1. For more details on the compared architectures, validation strategy and hyperparameter selection see App. C.1. For additional results concerning the training behavior, see App. C.3.

	IMDB-B	IMDB-M	RDT-B	RDT-M5K	COLLAB	MUTAG	PROTEINS	PTC	NCI1	AVG	p
GIN+SUM	71.8 ± 4.0	47.1 ± 4.3	85.5 ± 2.2	52.0 ± 3.0	70.9 ± 1.5	**87.2 ± 4.9**	**73.3 ± 3.1**	54.1 ± 7.1	**81.7 ± 2.3**	69.3	2.0e-5
GIN+MEAN	50.0 ± 0.0	33.3 ± 0.0	50.0 ± 0.1	20.0 ± 0.1	32.5 ± 0.1	76.1 ± 11.1	67.2 ± 2.9	58.7 ± 6.5	77.7 ± 1.9	51.7	3.2e-15
GIN+MAX	50.0 ± 0.0	33.3 ± 0.0	49.7 ± 0.5	20.2 ± 0.4	52.0 ± 0.0	77.0 ± 8.2	71.8 ± 3.6	59.0 ± 9.7	80.5 ± 2.8	54.8	3.9e-13
GIN+VPA	**72.0 ± 4.4**	**48.7 ± 5.2**	**89.0 ± 1.9**	**56.1 ± 3.0**	**73.5 ± 1.5**	86.7 ± 4.4	73.2 ± 4.8	**60.1 ± 5.8**	81.2 ± 2.1	**71.2**	-
GCN+SUM	63.3 ± 6.1	42.1 ± 3.7	75.4 ± 3.2	37.3 ± 3.5	67.0 ± 2.2	**78.7 ± 7.8**	70.3 ± 2.2	61.3 ± 7.8	**80.2 ± 2.0**	64.0	9.9e-9
GCN+MEAN	50.0 ± 0.0	33.3 ± 0.0	49.9 ± 0.2	20.1 ± 0.1	52.0 ± 0.0	72.4 ± 6.3	**74.3 ± 4.4**	**63.3 ± 6.5**	75.8 ± 2.6	54.6	3.3e-12
GCN+MAX	50.5 ± 0.0	33.3 ± 0.0	50.0 ± 0.0	20.0 ± 0.1	52.0 ± 0.0	67.6 ± 4.3	43.9 ± 7.3	58.7 ± 6.6	55.1 ± 2.6	47.8	1.4e-15
GCN+VPA	**71.7 ± 3.9**	**46.7 ± 3.5**	**85.5 ± 2.3**	**54.8 ± 2.4**	**73.7 ± 1.7**	76.1 ± 9.6	73.9 ± 4.8	61.3 ± 5.9	79.0 ± 1.8	**69.2**	-
SGC	62.9 ± 3.9	40.3 ± 4.1	78.9 ± 2.0	41.3 ± 3.5	68.0 ± 2.2	73.5 ± 9.8	73.1 ± 3.4	59.0 ± 6.0	68.5 ± 2.2	62.8	3.8e-12
SGC+VPA	**70.4 ± 4.1**	**47.5 ± 4.4**	**84.2 ± 2.2**	**53.4 ± 2.7**	**71.7 ± 1.7**	**73.9 ± 6.2**	**75.4 ± 4.2**	**63.1 ± 8.0**	**76.4 ± 2.8**	**68.4**	-
GAT	51.0 ± 4.4	37.4 ± 3.6	74.5 ± 3.8	33.1 ± 1.9	56.2 ± 0.6	77.7 ± 11.5	**75.4 ± 2.9**	60.5 ± 5.5	**77.7 ± 2.2**	60.4	7.6e-9
GAT+VPA	**71.1 ± 4.6**	**44.1 ± 4.5**	**78.1 ± 3.7**	**43.3 ± 2.4**	**69.9 ± 3.2**	**81.9 ± 8.0**	73.0 ± 4.2	**60.8 ± 6.1**	76.1 ± 2.3	**66.5**	-

Table 1: Test accuracy on the TUDatasets with 10-fold cross-validation. Standard deviations are indicated with ±. Column "AVG" shows the average test accuracy across data sets and column "p" indicates p-values of paired one-sided Wilcoxon tests across all datasets.

4 Discussion

Our results hint at a potentially powerful new aggregation function with equal expressivity as SUM aggregation and improved learning dynamics (for further details see Apps. A.3 and A.4). However, the results presented in this work are still limited to few datasets and architectures and require more extensive experimental testing.

[1]Note that this assumption is too strong, since for a fixed i, all $\boldsymbol{m}_{ij}$ depend on each other because they are all determined by the input $\boldsymbol{h}_i$

ACKNOWLEDGEMENTS

The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria. We thank the projects Medical Cognitive Computing Center (MC3), INCONTROL-RL (FFG-881064), PRIMAL (FFG-873979), S3AI (FFG-872172), DL for GranularFlow (FFG-871302), EPILEPSIA (FFG-892171), AIRI FG 9-N (FWF-36284, FWF-36235), AI4GreenHeatingGrids (FFG- 899943), INTEGRATE (FFG-892418), ELISE (H2020-ICT-2019-3 ID: 951847), Stars4Waters (HORIZON-CL6-2021-CLIMATE-01-01). We thank NXAI GmbH, Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Labs (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Verbund AG, GLS (Univ. Waterloo), Software Competence Center Hagenberg GmbH, Borealis AG, TÜV Austria, Frauscher Sensonic, TRUMPF and the NVIDIA Corporation.

URM STATEMENT

The authors acknowledge that at least one key author of this work meets the URM criteria of ICLR 2024 Tiny Papers Track.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer Normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message Passing Neural PDE Solvers. In *International Conference on Learning Representations*, 2022.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial Optimization and Reasoning with Graph Neural Networks. *Journal of Machine Learning Research*, 24(130):1–61, 2023.

Kaixuan Chen, Jie Song, Shunyu Liu, Na Yu, Zunlei Feng, Gengshi Han, and Mingli Song. Distribution Knowledge Embedding for Graph Pooling. *IEEE Transactions on Knowledge and Data Engineering*, 35(8):7898–7908, 2023.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal Neighbourhood Aggregation for Graph Nets. In *Advances in Neural Information Processing Systems*, 2020.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems*, 2016.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Advances in Neural Information Processing Systems*, 2015.

Ameen Eetemadi and Ilias Tagkopoulos. Genetic Neural Networks: An Artificial Neural Network Architecture for Capturing Gene Expression Relationships. *Bioinformatics*, 35(13):2226–2234, 11 2018.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph Neural Networks for Social Recommendation. In *The World Wide Web Conference*, 2019.

Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428*, 2019.

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*, 2017.

Xavier Glorot and Yoshua Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, 2010.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

Pieter-Jan Hoedt and Günter Klambauer. Principled Weight Initialisation for Input-Convex Neural Networks. In *Advances in Neural Information Processing Systems*, 2023.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, 2015.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular Graph Convolutions: Moving Beyond Fingerprints. *Journal of Computer-Aided Molecular Design*, 30 (8):595–608, 2016.

Ryan Keisler. Forecasting Global Weather with Graph Neural Networks. *arXiv preprint arXiv:2202.07575*, 2022.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-Normalizing Neural Networks. In *Advances in Neural Information Processing Systems*, 2017.

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Alexander Pritzel, Suman Ravuri, Timo Ewalds, Ferran Alet, Zach Eaton-Rosen, et al. GraphCast: Learning Skillful Medium-Range Global Weather Forecasting. *arXiv preprint arXiv:2212.12794*, 2022.

Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient BackProp. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller (eds.), *Neural Networks: Tricks of the Trade: Second Edition*, pp. 9–48. Springer Berlin Heidelberg, 2012.

Jaehoon Lee, Jascha Sohl-Dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep Neural Networks as Gaussian Processes. In *International Conference on Learning Representations*, 2018.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-Attention Graph Pooling. In *International Conference on Machine Learning*, 2019.

AA Leman and Boris Weisfeiler. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, 2(9):12–16, 1968.

Zhifei Li, Hai Liu, Zhaoli Zhang, Tingting Liu, and Neal N. Xiong. Learning Knowledge Graph Embedding With Heterogeneous Relation Attention Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(8):3961–3973, 2022.

Yao Lu, Stephen Gould, and Thalaiyasingam Ajanthan. Bidirectionally Self-Normalizing Neural Networks. *Neural Networks*, 167:283–291, 2023.

James Martens, Andy Ballard, Guillaume Desjardins, Grzegorz Swirszcz, Valentin Dalibard, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Rapid Training of Deep Neural Networks without Skip Connections or Normalization Layers using Deep Kernel Shaping. *arXiv preprint arXiv:2110.01765*, 2021.

Andreas Mayr, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K. Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. Large-Scale Comparison of Machine Learning Methods for Drug Target Prediction on ChEMBL. *Chemical Science*, 9:5441–5451, 2018.

Andreas Mayr, Sebastian Lehner, Arno Mayrhofer, Christoph Kloss, Sepp Hochreiter, and Johannes Brandstetter. Boundary Graph Neural Networks for 3D Simulations. In *AAAI Conference on Artificial Intelligence*, 2023.

Amil Merchant, Simon Batzner, Samuel S. Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. Scaling Deep Learning for Materials Discovery. *Nature*, 624(7990):80–85, 2023.

Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein. Fake News Detection on Social Media using Geometric Deep Learning. In *International Conference on Learning Representations*, 2019.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *AAAI Conference on Artificial Intelligence and Innovative Applications of Artificial Intelligence Conference and AAAI Symposium on Educational Advances in Artificial Intelligence*, 2019.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A Collection of Benchmark Datasets for Learning with Graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020.

Radford M Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, and Pascal Friederich. Graph Neural Networks for Materials Science and Chemistry. *Communications Materials*, 3(1):93, 2022.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to Simulate Complex Physics with Graph Networks. In *International Conference on Machine Learning*, 2020.

Sebastian Sanokowski, Wilhelm Franz Berghammer, Sepp Hochreiter, and Sebastian Lehner. Variational Annealing on Graphs for Combinatorial Optimization. In *Advances in Neural Information Processing Systems*, 2023.

Víctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) Equivariant Graph Neural Networks. In *International Conference on Machine Learning*, 2021.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. In *The Semantic Web*, 2018.

Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep Information Propagation. In *International Conference on Learning Representations*, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, 2018.

Juexin Wang, Anjun Ma, Qin Ma, Dong Xu, and Trupti Joshi. Inductive Inference of Gene Regulatory Network Using Supervised and Semi-Supervised Graph Neural Networks. *Computational and Structural Biotechnology Journal*, 18:3335–3343, 2020.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying Graph Convolutional Networks. In *International Conference on Machine Learning*, 2019.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, 2019.

APPENDIX

A SUPPLEMENTARY INFORMATION

A.1 EXTENDED ABSTRACT

Graph neural networks (GNNs), and especially message-passing neural networks are highly successful architectures for many application areas, such as physics, drug discovery, and molecular modeling. Message-passing neural networks generate messages for pairs of connected nodes and subsequently aggregate those messages via message aggregation functions. While the analogous aggregation step seems straightforward for vision-based convolutional neural networks (CNNs) due to a fixed number of input neurons per layer, the aggregation step in GNNs might be more delicate due to a variable number of neighbors per node. The type of aggregation function critically determines the expressivity of the neural network with respect to its ability to discriminate non-isomorphic graphs. While research has been focused on the expressivity of GNNs, there has been little attention to applying signal propagation theory to these networks. Under basic assumptions, signal propagation theory suggests a variance-preserving aggregation function that maintains expressivity, but yields improved forward and backward dynamics. Our experiments demonstrate that variance preservation allows faster training of GNNs and further we even observed improved prediction results on the respective test sets. Our results could pave the way towards normalizer-free or self-normalizing GNNs.

A.2 EXTENDED INTRODUCTION AND RELATED WORK

GNNs are useful for a variety of fields including molecule predictions (Duvenaud et al., 2015; Kearnes et al., 2016; Gilmer et al., 2017; Mayr et al., 2018; Satorras et al., 2021), material science (Reiser et al., 2022; Merchant et al., 2023), modelling physical interactions or improving PDE solvers for physics predictions (Sanchez-Gonzalez et al., 2020; Brandstetter et al., 2022; Mayr et al., 2023), weather prediction (Keisler, 2022; Lam et al., 2022), predictions about social networks (Hamilton et al., 2017; Fan et al., 2019; Monti et al., 2019), gene regulatory networks in systems biology (Eetemadi & Tagkopoulos, 2018; Wang et al., 2020), combinatorial optimization (Cappart et al., 2023; Sanokowski et al., 2023), and knowledge graphs (Schlichtkrull et al., 2018; Li et al., 2022) for reasoning.

Despite the huge successes of GNNs, there are some limitations. Morris et al. (2019) and Xu et al. (2019) analysed the expressive power of GNNs and found that they are not more powerful than the Weisfeiler-Leman graph isomorphism heuristic (1-WL test) (Leman & Weisfeiler, 1968) at distinguishing non-isomorphic graphs. Moreover, Xu et al. (2019) constructed a GNN (GIN architecture), which should attain the same expressive power as the 1-WL test. An important conclusion in the design of the GIN architecture was that SUM aggregation in the message aggregation step and the graph-level aggregation step allows to attain WL-1 expressive power, while MEAN or MAX aggregation effectively limit the expressive power.

The message aggregation step and the graph-level readout step are the critical operations in GNNs (Corso et al., 2020). Message passing on graphs involves the pair-wise exchange of messages, a message aggregation mechanism, and subsequent updates on nodes. This process can be linked to convolution operations (Wu et al., 2021; Kipf & Welling, 2017; Bronstein et al., 2021). However, unlike traditional convolutions, where the kernel size remains fixed, the message aggregation in GNNs is contingent on the number of neighboring nodes and, consequently, the incoming messages (Wu et al., 2021). A parallel rationale applies to graph-level readout operations[2], which can be grouped into topology-based pooling, hierarchical pooling, and global pooling (Lee et al., 2019). Advanced pooling mechanisms consider the graph as a distribution, from which nodes are sampled (Chen et al., 2023). Especially for graph-level readouts, similar operations are used as for message

[2]For graph-level readouts the distributed neural representation across the graph needs to be fused to a common representation space. This operation is denoted as pooling in the context of GNNs. For CNNs pooling often also refers also to the aggregation step itself. We will however be more strict in distinguishing aggregation and pooling here and consider pooling to be caused by the stride parameter of CNNs. Especially, for GNNs considered here, we are not interested in proposing a new pooling mechanism, but in suggesting a new aggregation function.

aggregation. Global pooling consolidates the graph information into a single hidden representation before making final predictions.

There are plenty of works on conventional fully-connected neural networks (FCNNs), which study signal propagation behavior (e.g., Schoenholz et al., 2017; Klambauer et al., 2017) throughout the networks. Typically, for FCNNs or CNNs, there are either weight initialisation schemes (e.g., Glorot & Bengio, 2010; He et al., 2015) or normalization layers (e.g., Ioffe & Szegedy, 2015; Ba et al., 2016), which avoid that the weighted summed inputs lead to exploding activations throughout the depth of the network. Exploding activations are a main obstacle in the efficient training of GNNs.

For GNNs and especially the GIN architecture with SUM message aggregation, signal propagation behavior appears problematic as well. Conventional weight initialization schemes at the aggregation steps can not be applied in a straightforward manner, since the number of neighbors in an aggregation step and the number of nodes in a graph are variable for each node. The fact, that zero variance in messages might be a common case for graph classification also limits the applicability of normalization layers.

Our aim in this work is to develop a general aggregation approach, which can be employed to different GNN architectures and which preserves maximum expressiveness and at the same time avoids exploding activations.

A.3 Extended Discussion

We compared VPA against other established aggregation methods on the same datasets as Xu et al. (2019) from the TUDataset benchmark (Morris et al., 2020).

In general it needs to be considered that better prediction performance of expressively more powerful GNNs will only be observed, when the underlying machine learning problem requires such a level of expressiveness to be solved well. For benchmarks from real-world data it might however not be known, whether expressively less powerful GNNs could also show competitive prediction performance.

Furthermore, variance preservation seems to be an important property to avoid exploding or vanishing activations. This is especially relevant for very deep networks. For the datasets used, all methods could be trained without diverging due to exploding activations. One reason for this could be that the GNNs are quite shallow and therefore there are only few message passing steps. Nevertheless, learning curves shown in Fig. C1 suggest that VPA has advantages over SUM aggregation in terms of convergence speed.

On the social network datasets, VPA seems to perform particularly well compared to other methods when no additional node features are introduced, forcing the GNNs to learn from the network structure instead (see experimental results in Table 1). However, including the node degree as a feature improves the performance of less expressive GNNs (see Table C1). The advantage in prediction performance of VPA over other methods is less pronounced in this setting.

A.4 Limitations

We suggest VPA as a general aggregation scheme. For some architectures, such as GINs, it is straightforward how to use VPA within the respective architecture. For other methods, however, its application might not be obvious. For example, SGC inherently contains a normalization strategy using node degrees and GAT makes use of attention weights during aggregation. In both cases, signal propagation is affected. Taking this into account, we suggest variants of VPA for SGC and GAT. Variance preservation for GAT+VPA is shown in Lemma 3, however, we did not formally prove variance preservation for SGC+VPA.

It should further be considered, that distributional assumptions to formally show variance preservation for VPA might only hold at the time of initialization. However, as discussed in App. B.1 even that time point is important. Furthermore, even under other assumptions on the distribution of the messages, arguments about increase and decrease of variance would hold.

B TECHNICAL DETAILS & FURTHER RESULTS

B.1 MLP SIGNAL PROPAGATION

In accordance with signal propagation literature (Schoenholz et al., 2017; Klambauer et al., 2017) we are interested in signal propagation of randomly initialized neural networks, i.e., we assume distributions on weights of these networks. Although it might also seem interesting to know about signal propagation behavior at different time points during training, this is much more difficult to study, since the distributions of weights might then also depend on the training data. An argumentation for studying signal propagation at initialization however would be, that learning might not work at all (or start well), when signal propagation throughout the whole network does not even work (well) at initialization.

In order to investigate the forward dynamics of a message-passing network at initialization time with signal propagation theory, we take the following assumptions, assuming the case of ϕ taking two arguments[3]. The initial representation of pairs of node representations $\boldsymbol{h}_{ij}^P = (\boldsymbol{h}_i, \boldsymbol{h}_j)$ with $i \neq j$ follows a data distribution $\boldsymbol{h}_{ij}^P \sim p_{\text{data}}$ with some mean $\mathrm{E}_{\boldsymbol{h}^P \sim p_{\text{data}}}\left(\boldsymbol{h}_{ij}^P\right) = \boldsymbol{\mu}_{\boldsymbol{h}^P}$ and some covariance $\mathrm{Cov}_{\boldsymbol{h}^P \sim p_{\text{data}}}\left(\boldsymbol{h}_{ij}^P\right) = \boldsymbol{C}_{\boldsymbol{h}^P}$.

We further assume a deep and broad MLP $\phi_{\boldsymbol{w}}(.)$ with randomly sampled weights according to LeCun's initialization scheme (LeCun et al., 2012), $w \sim p_{\mathcal{N}}(0, 1/H)$, where H is the fan-in of each neuron, and with linear activation in the last layer. Since an MLP ϕ is a measurable function, $\boldsymbol{m}_{ij} = \phi_{\boldsymbol{w}}(\boldsymbol{h}_i, \boldsymbol{h}_j)$ is also a random variable. Then central results from signal propagation theory (Neal, 1995; Schoenholz et al., 2017; Lee et al., 2018; Hoedt & Klambauer, 2023) imply that the distribution of $\boldsymbol{m}_{ij}$ at initialization can be approximated by a standard normal distribution $\boldsymbol{m}_{ij} \sim p_{\mathcal{N}}(\boldsymbol{0}, \boldsymbol{I})$ (Lee et al., 2018, Section 2.2) and even a fixed point at zero mean and unit variance can be enforced (Klambauer et al., 2017; Lu et al., 2023). In practice, batch- (Ioffe & Szegedy, 2015) or layer-norm (Ba et al., 2016) are often used in these MLPs to partly maintain these statistics, i.e. zero mean and unit variance, also during learning. We are aware that this approximation only holds at initialization and might be overly simplistic (Martens et al., 2021) (see Section 4). However, note that we use this assumption only to make the point of variance preservation of the aggregation step. Even under other assumptions on the distribution of $\boldsymbol{m}_{ij}$ the arguments about increase and decrease of variance would hold.

B.2 VARIANCE PRESERVATION

Lemma B1. *Let $z_1, \ldots, z_N$ be independent copies of a centered random variable z with finite variance. Then the random variable $y = \frac{1}{\sqrt{N}} \sum_{n=1}^{N} z_n$ has the same mean and variance as z.*

Proof. Because the variables z_n are centered, we have

$$\mathrm{E}[y] = \mathrm{E}\left[\frac{1}{\sqrt{N}} \sum_{n=1}^{N} z_n\right] = \frac{1}{\sqrt{N}} \sum_{n=1}^{N} \mathrm{E}[z_n] = 0 = \mathrm{E}[z]. \tag{B1}$$

Furthermore, we have

$$\mathrm{Var}[y] = \mathrm{E}\left[\left(\frac{1}{\sqrt{N}} \sum_{n=1}^{N} z_n\right)^2\right] - \mathrm{E}\left[\frac{1}{\sqrt{N}} \sum_{n=1}^{N} z_n\right]^2 = \tag{B2}$$

$$= \mathrm{E}\left[\frac{1}{N}\left(\sum_{n=1}^{N} z_n\right)^2\right] = \frac{1}{N}\mathrm{E}\left[\sum_{n=1}^{N} z_n^2 + \sum_{n=1}^{N} \sum_{m=1, m\neq n}^{N} 2 z_n z_m\right] = \tag{B3}$$

$$= \frac{1}{N} N \mathrm{E}[z_n^2] = \mathrm{Var}[z_n] = \mathrm{Var}[z], \tag{B4}$$

[3]For the one-argument version of ϕ (where the message is computed only from the node representation $\boldsymbol{h}_j$ of the neighboring node) the line of reasoning is almost analogous.

where we have used the independence assumption $\mathrm{E}[z_n z_m] = \mathrm{E}[z_n]\mathrm{E}[z_m] = 0$ and that the z_n are centered, which means that $\mathrm{E}[z_n^2] = \mathrm{Var}[z_n]$.

□

B.3 EXPRESSIVITY

Lemma B2. *Assume the multiset $\mathcal{X}$ is countable and the number of unique elements in $\mathcal{X}$ is bounded by a number N. There exists a function $f : \mathcal{X} \to \mathbb{R}^N$ such that $h(X) = \frac{1}{\sqrt{|X|}} \sum_{x \in X} f(x)$ is unique for each multiset $X \subset \mathcal{X}$ of bounded size, where $|X|$ denotes the cardinality of multiset X (sum of multiplicities of all unique elements in the multiset).*

Proof. Since the number of unique elements in $\mathcal{X}$ is bounded by N, there exists a bijective mapping $Z : \mathcal{X} \to \{1, ..., N\}$ assigning a natural number to each $x \in \mathcal{X}$. Then an example of such a function f is a one-hot encoding function $f(x) = e_{Z(x)}$, with $e_{Z(x)} \in \mathbb{R}^N$ being a standard basis vector, where component i of $e_{Z(x)}$, i.e. $e_{Z(x)}[i]$ is defined as $e_{Z(x)}[i] := \begin{cases} 0 & \text{for} \quad i \neq Z(x) \\ 1 & \text{for} \quad i = Z(x) \end{cases}$.

We define $h(X)$ to be:

$$h(X) = \frac{1}{\sqrt{|X|}} \sum_{x \in X} f(x) = \frac{1}{\sqrt{|X|}} \sum_{x \in X} e_{Z(x)}.$$

Summing up the components of $h(X)$ yields the square root of the cardinality of X, i.e. the embeddings contains information on the cardinality of X. Since we know, $\sqrt{|X|}$ from the embedding, we can just multiply the embedding $h(X)$ with $\sqrt{|X|}$ to obtain the original multiplicity of each element x in multiset X. Thus, the multiset $X \subset \mathcal{X}$ can be uniquely reconstructed from $h(X)$, implying that h is injective.

□

We note, that for MEAN aggregation, i.e., $\tilde{h}(X) = \frac{1}{|X|} \sum_{x \in X} f(x)$, the multiset X cannot be reconstructed from $\tilde{h}(X)$, since in that case the components of $\tilde{h}(X)$ sum up to 1 and therefore, do not indicate the cardinality of X (e.g., $h(\{0, 1\}) = (0.5, 0.5) = h(\{0, 0, 1, 1\})$). In contrast, for $h(X) = \frac{1}{\sqrt{|X|}} \sum_{x \in X} f(x)$, the embeddings contain information on the cardinality of X, which is lost for MEAN aggregation. Multiplication by $|X|$ does not work for MEAN aggregation to reconstruct the original multiset X, as no cardinality information is stored in the embedding $\tilde{h}(X)$. More generally, no function f can be found such that $\tilde{h}(X)$ is unique for each multiset $X \subset \mathcal{X}$ of bounded size (see Corollary 8 in Xu et al. (2019)).

B.4 EXTENSION OF VARIANCE PRESERVATION TO ATTENTION

In the following, we show, how a variance-preserving aggregation strategy could be extended to attention mechanisms. Although we do empirically investigate this strategy, we suggest that it can be employed as part of graph attention networks and therefore consider it as future work.

We assume, that random variables $z_1, \ldots, z_N$ are aggregated by an attention mechanism. The respective computed attention weights are assumed to be given by $c_1, \ldots, c_N$, where $c_i \in \mathbb{R}_0^+$ and $\sum_{i=1}^N c_i = 1$ holds. Further, we consider c_i to be constants [4].

In order to find a useful extension of VPA to attention, we first consider two extreme cases on the distribution of attention weights:

- Case 1: All attention weights are equal. Then, in order to fulfill $\sum_{i=1}^N c_i = 1$, all $c_i = \frac{1}{N}$.

[4]Note, that this might be an over-simplistic assumption, especially since/when keys and values are not independent.

- Case 2: Attention focuses on exactly one value, which might be w.l.o.g. j. Then $c_j = 1$ and $c_i = 0 \ \forall \ i \neq j$.

We note, that case 1 is the same as MEAN aggregation and case 2 is the same as MAX aggregation if $max(z_1, \ldots, z_N) = z_j$ and $z_i < z_j \ \forall \ i \neq j$. For both cases, GNNs have more limited expressivity than with VPA or SUM aggregation.

A first extension, to increase expressivity for case 1, could therefore be, to change the attention mechanism from $y = \sum_{i=1}^{N} c_i \ z_i$ to $y = \sqrt{N} \ \sum_{i=1}^{N} c_i \ z_i$. Then case 1 is analogous to VPA: $y = \sqrt{N} \ \sum_{i=1}^{N} \frac{1}{N} \ z_i = \frac{1}{\sqrt{N}} \ \sum_{i=1}^{N} \ z_i$.

A more general extension of variance preservation to attention might be to compute a constant $C = \sqrt{\sum_{i=1}^{N} c_i^2}$ and use the following attention mechanism: $y = \frac{1}{C} \ \sum_{i=1}^{N} c_i \ z_i$

Lemma B3. *Let $z_1, \ldots, z_N$ be independent copies of a centered random variable z with finite variance and let $c_1, \ldots, c_N$ be constants, where $c_i \in \mathbb{R}_0^+$ and $\sum_{i=1}^{N} c_i = 1$. Then the random variable $y = \frac{1}{C} \sum_{n=1}^{N} c_n z_n$ with $C = \sqrt{\sum_{i=1}^{N} c_i^2}$ has the same mean and variance as z.*

Proof. Because the variables z_n are centered, we have

$$\mathrm{E}[y] = \mathrm{E}\left[\frac{1}{C}\sum_{n=1}^{N} c_n z_n\right] = \frac{1}{C}\sum_{n=1}^{N} c_n \mathrm{E}[z_n] = 0 = \mathrm{E}[z]. \tag{B5}$$

Furthermore, we have

$$\mathrm{Var}[y] = \mathrm{E}\left[\left(\frac{1}{C}\sum_{n=1}^{N} c_n z_n\right)^2\right] - \mathrm{E}\left[\frac{1}{C}\sum_{n=1}^{N} c_n z_n\right]^2 = \tag{B6}$$

$$= \mathrm{E}\left[\frac{1}{C^2}\left(\sum_{n=1}^{N} c_n z_n\right)^2\right] = \frac{1}{C}\mathrm{E}\left[\sum_{n=1}^{N} c_n^2 z_n^2 + \sum_{n=1}^{N}\sum_{m=1, m\neq n}^{N} 2 c_n c_m z_n z_m\right] = \tag{B7}$$

$$= \frac{1}{C^2}\sum_{n=1}^{N} c_n^2 \mathrm{E}[z_n^2] = \frac{1}{\sum_{i=1}^{N} c_i^2}\Big(\sum_{i=1}^{N} c_i^2\Big)\mathrm{E}[z_n^2] = \mathrm{Var}[z_n] = \mathrm{Var}[z], \tag{B8}$$

where we have used the independence assumption $\mathrm{E}[z_n z_m] = \mathrm{E}[z_n]\mathrm{E}[z_m] = 0$ and that the z_n are centered, which means that $\mathrm{E}[z_n^2] = \mathrm{Var}[z_n]$.

□

Note, that for case 1 (uniform attention weights), $C = \sqrt{\sum_{i=1}^{N} c_i^2} = \sqrt{\sum_{i=1}^{N} \left(\frac{1}{N}^2\right)} = \sqrt{N \frac{1}{N^2}} = \frac{1}{\sqrt{N}}$. Further $y = \frac{1}{C} \ \sum_{i=1}^{N} c_i \ z_i = \frac{1}{\frac{1}{\sqrt{N}}} \ \sum_{i=1}^{N} \frac{1}{N} \ z_i = \sqrt{N} \ \frac{1}{N} \sum_{i=1}^{N} \ z_i = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} \ z_i$ is obtained, which is the same as VPA.
Case 2, i.e., $c_j = 1$ and $c_i = 0 \ \forall \ i \neq j$ gives $C = 1$, and $y = \frac{1}{C} \ \sum_{i=1}^{N} c_i \ z_i = z_j$. Cardinality information is lost in this case. However, the attention mechanism might be learnable and therefore not converge to this solution if limited expressivity leads to larger losses during optimization.

C EXPERIMENTAL DETAILS & FURTHER RESULTS

C.1 IMPLEMENTATION DETAILS

GNN architectures and aggregation functions compared. The models used in our experiments were Graph Isomorphism Networks (GIN) (Xu et al., 2019), Graph Convolutional Network (GCN) (Kipf & Welling, 2017), Graph Attention Networks (GAT) (Veličković et al., 2018) and Simple Graph Convolution Networks (SGC) (Wu et al., 2019). To evaluate prediction performance, we combined GIN and GCN architectures with each of the aggregation methods in Fig. 1 - sum aggregation (SUM), mean aggregation (MEAN), max aggregation (MAX) and variance- preserving aggregation (VPA) - both for message aggregation and graph-level readout. Note that we used the GCN formulation as reported in Morris et al. (2019) to circumvent the inherent normalization in the GCN architecture by Kipf & Welling (2017).

To incorporate the idea of variance preservation into the SGC architecture, we changed the update of $\mathbf{h}$ from

$$\mathbf{h}_i' = \frac{1}{d_i + 1}\mathbf{h}_i + \sum_{j=1}^{N} \frac{a_{ij}}{\sqrt{(d_i + 1)(d_j + 1)}}\mathbf{h}_j$$

to

$$\mathbf{h}_i' = \frac{1}{\sqrt{d_i + 1}}\mathbf{h}_i + \sum_{j=1}^{N} \frac{a_{ij}}{\sqrt[4]{(d_i + 1)(d_j + 1)}}\mathbf{h}_j$$

(where a_{ij} are entries of the adjacency matrix, d_i and d_j are node degrees, and, $\mathbf{h_i}$ and $\mathbf{h_j}$ denote the hidden neural representation at some time step during message passing. For a variance-preserving version of GAT, we suggest an extension as presented in App. B.4. More specifically, we employ the normalization constant suggested in Lemma B3 and note, that in a practical implementation we do not backpropagate errors through these constants during training.

Hyperparameters. We extended our framework upon implementations as provided by PyTorch Geometric (Fey & Lenssen, 2019). Specifically, we used the following convolutional layers: GINConc (GIN), GraphConv (GCN), SGConv (SGC) and GATConv (GAT). We used 5 GNN layers for GIN, GCN and GAT, respectively, and one layer with $K = 5$ hops for SGC. The dimension of the messages was 64 for all architectures. An MLP with one hidden layer was used for classification with a hidden dimension of 64 for GIN and 128 for all other models. We used a dropout-rate of 0.5 and the standard Adam optimizer with a learning rate of 0.001.

Benchmarking datasets and settings. We tested our methods on the same graph classification benchmarks from the TUDataset collection as Xu et al. (2019), consisting of five social network datasets (IMDB-BINARY, IMDB-MULTI, COLLAB, REDDIT-BINARY, and REDDIT-MULTI-5K) and four bioinformatics datasets (MUTAG, PROTEINS, PTC and NCI1). Since the social datasets do not contain any node features, we introduced node features in two different ways. In the first variant the graphs are considered as given with all node features set to 1. In the other variant the one-hot encoded node degree is used as an additional node feature. We report results for the first variant in Table 1 and results for the second variant in Table C1. The bioinformatics datasets were used with the provided node features. For more details on the used datasets, we refer to Morris et al. (2020) and Xu et al. (2019).

Training, validation, and test splits. Our experiments were evaluated with 10-fold cross-validation. In each iteration, we used $1/10$ of the data as test set, $1/10$ for validation and $8/10$ for training. The validation set was only used to adjust the number of training epochs, such that our test accuracies were computed for the epoch with the highest validation accuracy. For other hyperparameters we made use of default settings in compared methods.

Code availability. The code used to produce our results is available at https://github.com/ml-jku/GNN-VPA. Moreover, VPA has been integrated into the PyTorch Geometric framework as an aggregation function and will be released in version

2.6.0 (https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.aggr.VariancePreservingAggregation.html).

C.2 EXTENDED RESULTS

Table C1 shows results for the social datasets from the benchmark datasets used by Xu et al. (2019) with preprocessed graphs. Please refer to App. A.3 for a discussion about these results compared to those in Table 1.

	IMDB-B	IMDB-M	RDT-B	RDT-M5K	COLLAB
GIN+SUM	72.5 ± 4.5	$\mathbf{50.8 \pm 4.1}$	81.5 ± 1.7	$\mathbf{47.5 \pm 2.4}$	$\mathbf{82.2 \pm 1.7}$
GIN+MEAN	$\mathbf{73.8 \pm 4.4}$	48.9 ± 3.7	77.1 ± 2.8	47.1 ± 1.6	80.7 ± 1.0
GIN+MAX	71.0 ± 4.5	47.5 ± 4.9	78.5 ± 2.2	42.7 ± 2.1	77.1 ± 1.7
GIN+VPA	73.7 ± 3.7	49.7 ± 3.6	$\mathbf{82.0 \pm 2.0}$	47.4 ± 1.9	$\mathbf{82.2 \pm 1.7}$
GCN+SUM	70.7 ± 3.1	43.9 ± 3.7	76.3 ± 3.6	$\mathbf{50.4 \pm 2.4}$	73.7 ± 2.2
GCN+MEAN	71.9 ± 5.2	$\mathbf{51.3 \pm 3.4}$	71.0 ± 2.5	46.3 ± 2.3	80.6 ± 1.0
GCN+MAX	62.9 ± 3.5	43.1 ± 4.2	63.4 ± 5.0	30.6 ± 2.6	74.8 ± 1.6
GCN+VPA	$\mathbf{73.6 \pm 5.5}$	50.5 ± 2.7	$\mathbf{80.6 \pm 3.4}$	47.9 ± 2.3	$\mathbf{81.3 \pm 1.5}$
SGC	$\mathbf{72.9 \pm 3.9}$	$\mathbf{50.6 \pm 3.5}$	81.0 ± 2.4	$\mathbf{49.0 \pm 1.9}$	$\mathbf{81.3 \pm 1.8}$
SGC+VPA	72.6 ± 3.7	49.4 ± 3.6	$\mathbf{81.5 \pm 2.3}$	47.8 ± 2.8	80.5 ± 1.1
GAT	$\mathbf{73.9 \pm 3.4}$	$\mathbf{50.2 \pm 4.0}$	78.3 ± 3.0	47.0 ± 2.7	$\mathbf{81.2 \pm 1.4}$
GAT+VPA	71.7 ± 4.9	49.6 ± 6.1	$\mathbf{79.1 \pm 2.3}$	$\mathbf{47.5 \pm 1.7}$	79.5 ± 1.5

Table C1: Results on the social datasets of the benchmark setting by (Xu et al., 2019). In this variant of the datasets, the number of neighbors of a node is encoded as a node feature. The compared methods are again GIN and GCN with four different aggregation functions and SGC and GAT with their tailor-made variance preservation modifications.

C.3 LEARNING DYNAMICS

We investigated the learning dynamics of the compared methods based on the training loss curves (see Figure C1). The learning curves show that GIN model training converges fast with MEAN, MAX and VPA and slower with SUM aggregation, which we attribute to the exploding variance in the forward pass.

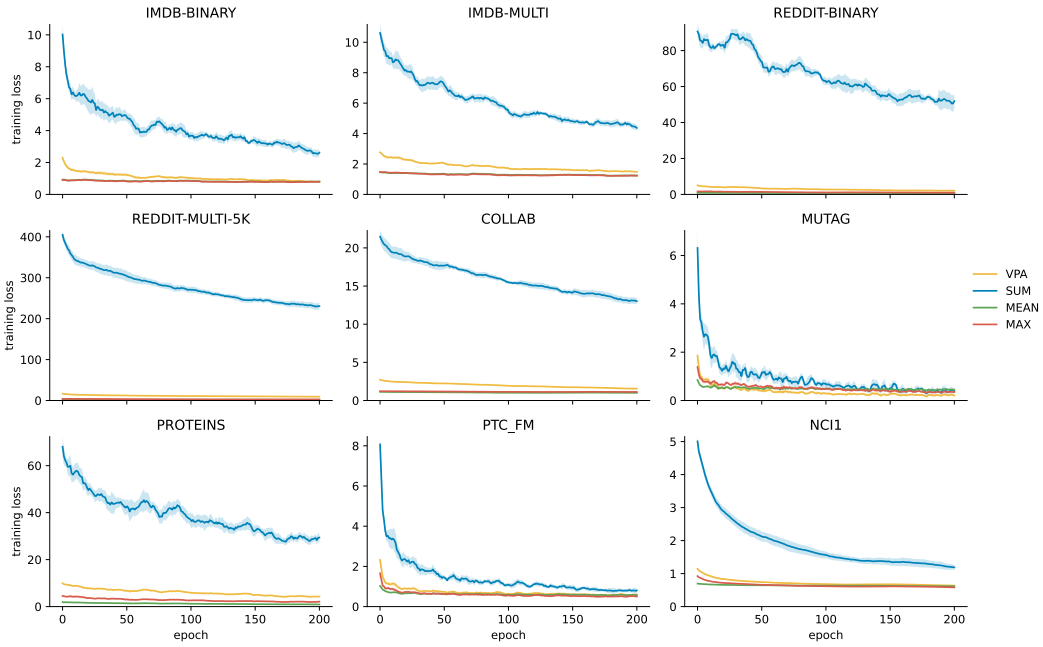

Figure C1: Learning Curves of the GIN architecture with different aggregation functions on the TUDataset benchmarks used by Xu et al. (2019) and which were retrieved in the version as provided by Morris et al. (2020). Note that the default hyperparameters are adjusted to the SUM aggregation function. Nevertheless, the network training converges faster with variance-preserving aggregation (VPA) compared to SUM aggregation. At the same time, VPA also maintains expressivity, whereas MEAN and MAX aggregation decrease the expressivity of GNNs.