# The Self-Loop Paradox: Investigating the Impact of Self-Loops on Graph Neural Networks

**Moritz Lampert, Ingo Scholtes**
Chair of Machine Learning for Complex Networks
Center for Artificial Intelligence and Data Science (CAIDAS)
Julius-Maximilians-Universität Würzburg, DE
moritz.lampert@uni-wuerzburg.de

## Abstract

Many Graph Neural Networks (GNNs) add self-loops to a graph to include feature information about a node itself at each layer. However, if the GNN consists of more than one layer, this information can return to its origin via cycles in the graph topology. Intuition suggests that this "backflow" of information should be larger in graphs with self-loops compared to graphs without. In this work, we counter this intuition and show that for certain GNN architectures, the information a node gains from itself can be smaller in graphs with self-loops compared to the same graphs without. We adopt an analytical approach for the study of statistical graph ensembles with a given degree sequence and show that this phenomenon, which we call the *self-loop paradox*, can depend both on the number of GNN layers $k$ and whether $k$ is even or odd. We experimentally validate our theoretical findings in a synthetic node classification task and investigate its practical relevance in 23 real-world graphs.

## 1 Introduction

Multi-layer Perceptrons (MLPs) use feature information of a sample to make predictions. GNNs use the sample represented as a node in a graph and feature information of the neighbouring nodes for predictions. To include the information about a node itself, GNN architectures like Graph Convolutional Network (GCN) [8] add self-loops. For single-layer GNNs, it is necessary to include the node's feature but for more than one layer, information can also flow back via cycles in the topology of the graph. Intuitively, adding self-loops still increases the information a node learns about itself. Thus, the influence of self-loops on GNNs remains mainly unexplored up to now.

Recent works investigate the influence of a graph's topology on GNNs in combination with over-squashing, i.e. GNNs perform badly on tasks that rely on interactions between distant nodes. The exponentially growing number of neighbours with increasing distance is one reason for over-squashing [1]. The GNNs perform badly because useful information of distant nodes is squashed by bottlenecks in the graph topology [5, 14] and the redundancy introduced by close neighbours [2]. The works on over-squashing all have in common that they connect the graph topology via walks between nodes to the influence nodes have on each other's predictions in a GNN. Earlier work [18] uses a similar approach in a different context and also mentions how self-loops can influence GNN predictions. Topping et al. [14] build on that idea and investigate it further.

We build upon the concept that the influence of a node feature on the prediction of another node is proportional to the relative number of walks between them out of all walks. In particular, we investigate the proportion of cycles of length $k$ from a node $v$ out of all walks of length $k$ ending in $v$. Using theory from network science based on random graphs with a given degree sequence [10, 12], we prove that including self-loops in those random graphs

M. Lampert et al., The Self-Loop Paradox: Investigating the Impact of Self-Loops on Graph Neural Networks (Extended Abstract). Presented at the Second Learning on Graphs Conference (LoG 2023), Virtual Event, November 27–30, 2023.

decreases the proportion of cycles with length 2. We further show that this can reduce a node's influence on its own prediction in two-layered GNNs. The theoretical findings are validated empirically using synthetic data generated using a Stochastic Block Model (SBM) [7] and put into context on real-world datasets in the appendix.

## 2 Background

**Random Graphs.** For the statistical analysis, we adopt the probabilistic Molloy-Reed configuration model [11] that generates random undirected graphs $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ based on a given degree sequence $S = (d_v)_{v \in \mathbf{V}}$. Under this model, if the degree sequence $S$ is graphic, each graph $\mathcal{G}$ with the same degree sequence $S$ is equiprobable. Computationally, this can be achieved by adding $d_v$ edge stubs for each node $v$ with degree $d_v$ and then connecting pairs of randomly chosen stubs until no stubs are left.
While it is easy to algorithmically generate random graphs based on the Molloy-Reed model, its true importance in network science is due to our ability to *analytically* study expected properties of these random graphs based on generating functions [12, 16]. For this type of model, we can express the expected degree of a randomly chosen node as

$$\langle d \rangle = \frac{1}{|\mathbf{V}|} \sum_{d_v \in \mathbf{S}} d_v. \tag{1}$$

The expected degree of a random neighbour of a randomly chosen node is given by the expected degree and its second raw moment: [12]

$$\langle d_N \rangle = \frac{\langle d^2 \rangle}{\langle d \rangle} \tag{2}$$

The number of walks of length $k$ from node $u$ to $v$ can be computed using the adjacency matrix $A$ of graph $\mathcal{G}$ as $A^k_{uv}$. The expected total number of $k$-length walks from or to $v$ is given by [12]

$$\mathbb{E}\left(\sum_{u \in \mathbf{V}} A^k_{uv}\right) = \mathbb{E}\left(\sum_{u \in \mathbf{V}} A^k_{vu}\right) = \langle d \rangle \cdot \langle d_N \rangle^{k-1} = \langle d \rangle \cdot \left(\frac{\langle d^2 \rangle}{\langle d \rangle}\right)^{k-1}. \tag{3}$$

**Graph Neural Networks.** Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a graph without self-loops, with $n$ nodes, $m$ edges and adjacency matrix $A$. Then given the input features of the previous layer $\mathbf{h}_u^{(k-1)}$, a GCN-layer [8] is defined as

$$\mathbf{h}_v^{(k)} = \sigma^{(k)}\left(\left(\sum_{u \in N(v) \cup \{v\}} \frac{1}{\sqrt{d(v)d(u)}} \cdot \mathbf{h}_u^{(k-1)}\right) \cdot W^{(k)} + b^{(k)}\right), \tag{4}$$

where $N(v)$ yields the set of neighbours of $v$ and $d(v)$ is the degree of $v$. The weight matrix $W^{(k)}$ and the bias vector $b^{(k)}$ are trainable parameters of the $k$-th layer and $\sigma^{(k)}$ is a non-linear activation function.
Note that the feature $\mathbf{h}_v^{(k)}$ of the node $v$ itself is included in the summation without any special treatment. This is the same as adding self-loops to the graph topology and using the adjacency matrix $\tilde{A} = A + I$. Other GNNs like GraphSAGE [6] or the Graph Isomorphism Network (GIN) [17] treat the node's feature differently and, thus, the theoretical findings of this work do not directly apply to these GNNs (see Appendix D).
To measure the influence of an input feature on the prediction of a node $v$, we use the Jacobian of the output with respect to a certain input feature and make use of a finding by Chen et al. [2] in the context of over-squashing for Message Passing GNNs (MPNNs) [4]:
**Lemma 1.** *If the MPNN passes messages along all $k$-length walks from $u$ to $v$ with equal probability, then the relative influence of input feature $\mathbf{h}_u^{(0)}$ on node output $\mathbf{h}_v^{(k)}$ is on average*

$$\mathbb{E}\left(\frac{\partial \mathbf{h}_v^{(k)} / \partial \mathbf{h}_u^{(0)}}{\sum_{u' \in V} \partial \mathbf{h}_v^{(k)} / \partial \mathbf{h}_{u'}^{(0)}}\right) = \frac{\tilde{A}^k_{uv}}{\sum_{u' \in V} \tilde{A}^k_{u'v}}. \tag{5}$$

This means that it is possible to make statements about the influence of specific input features on the prediction by analysing the graph topology and the number of walks of a certain length between two nodes.

## 3 The Self-Loop Paradox

**Theoretical Analysis.** We prove that relatively speaking, more walks lead back to the node itself in graphs without self-loops compared to graphs with self-loops:

**Lemma 2.** *Given a graph $\mathcal{G}$ generated using the configuration model [11] with adjacency matrix $A$ without self-loops and its counterpart with self-loops $\tilde{\mathcal{G}}$ with $\tilde{A} = A + I$, the proportion of cycles of length 2 from a node $v$ to itself out of all walks of length 2 ending in $v$ is larger in $\mathcal{G}$ than in $\tilde{\mathcal{G}}$:*

$$\frac{\mathbb{E}\left(A_{vv}^2\right)}{\mathbb{E}\left(\sum_{u \in V} A_{uv}^2\right)} > \frac{\mathbb{E}\left(\tilde{A}_{vv}^2\right)}{\mathbb{E}\left(\sum_{u \in V} \tilde{A}_{uv}^2\right)}. \tag{6}$$

We prove Lemma 2 in Appendix A and experimentally validate the given proportions on example graphs in Appendix B. This means in combination with Lemma 1 that including self-loops can decrease the information a node retains about itself in certain MPNNs with two layers. A detailed discussion on which GNNs are affected is provided in Appendix D. This is counter-intuitive since for single-layer GNNs, the node's feature is only included via self-loops. Although strictly speaking, this only applies to graphs generated from the configuration model, we show that this effect can also be observed in other random graphs below and real-world networks in Appendix C. This is in line with other properties of the configuration model that are also observed in real-world networks [13, pp. 377–381].

**Empirical Validation.** To empirically validate our theoretical results on GNNs, we use a GCN model to address a node classification task in synthetic graphs generated based on the Stochastic Block Model (SBM) [7]. This model generates random undirected graphs with $c$ classes with a label $y_v$ for each node. A stochastic block matrix $P \in (0,1)^{c \times c}$ defines the probability $p_{ij}$ that an edge between a pair of nodes with class $i$ and $j$ exists.
We use the SBM to generate graphs with ten classes and 100 nodes per class. For each class, we sample 16-dimensional node features from a Normal distribution centred around different corners of a unit hypercube with standard deviation $\sigma = 0.4$. Edge probabilities $p_{ii}$ (between nodes with the same class) and $p_{ij}$ (between nodes in different classes) are reported in Fig. 1. The generated graphs used for results in the top left figure ($p_{ii} > p_{ij}$) exhibit strong cluster patterns, while graphs in the bottom right figure ($p_{ii} = p_{ij}$) exhibit no cluster patterns.
For graphs with strong cluster patterns, a GCN can achieve high accuracy in a node classification task based on the graph topology alone, while node features play a subordinate role. On the contrary for graphs with no strong cluster patterns, the features of nodes play a more pronounced role. This should highlight the influence of self-loops on the performance of GCNs with different numbers of layers. Based on our theoretical findings, we thus expect that the performance difference of GCNs with and without self-loops shows a dependency on the number of layers $k$, where we expect models without self-loops to outperform those with self-loops for $k = 2$ layers, while we expect the opposite behaviour for $k = 1$.

The GCNs are trained with 80% of the nodes for 70 epochs using an Adam optimizer with learning rate 0.01. Fig. 1 shows the accuracies of the trained GCNs on the remaining nodes. We test different numbers of layers $k \in \{1, \ldots, 6\}$ for the GCN architecture and average the results over 50 runs. The code of our experiments is available on GitHub[1].
The results show that the difference between the accuracies of GCNs with and without self-loops varies with the number of layers $k$. As expected, with increasing "noise" in the topology, i.e. less pronounced cluster patterns, the general performance of a GCN decreases, eventually achieving a performance that is lower than that of an MLP that exclusively uses node features (grey line in Fig. 1). Moreover, the results suggest that –for graphs with a high level of topological noise– the performance of a GCN strongly depends on the question of whether the number of layers is odd or even. In line with our theoretical analysis, our results suggest that the inclusion of self-loops for $k = 2$ layers squashes the nodes' own features. For graphs with the highest level of "noise" self-loops actually harm the performance of a two-layered GCN architecture as opposed to a single-layer architecture.
In Appendix C, we further test our theoretical findings using 23 empirical benchmark graphs

[1] https://github.com/M-Lampert/self-loop-paradox

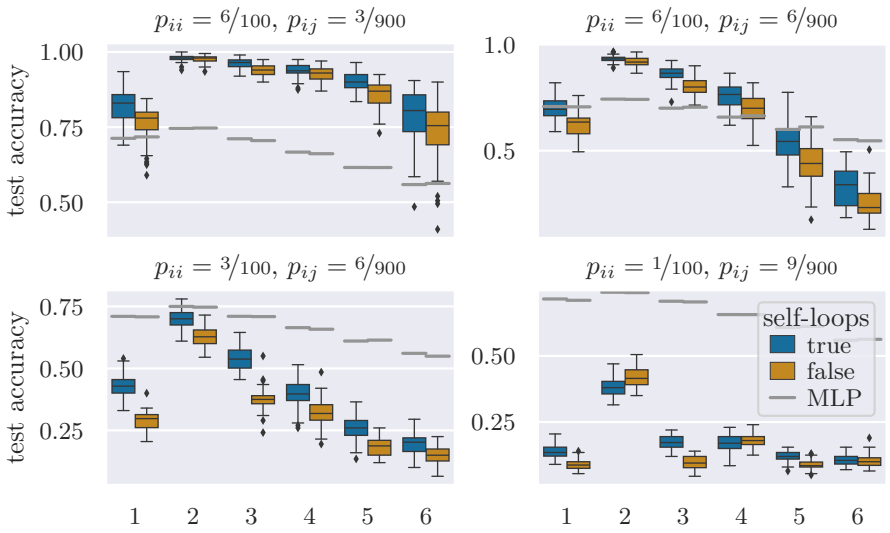


**Figure 1:** Test set accuracies of GCNs with and without self-loops averaged over 50 runs for different numbers of layers. Coloured boxes mark the range between the first and third quartiles. Whiskers stretch to the furthest outliers within three halves of that range while other outliers outside of the whiskers are marked with a rhombus. Grey lines show average accuracies of MLPs with the corresponding number of layers.

from PyTorch Geometric [3]. The results in Tab. 3 show that the self-loop paradox indeed holds for 13 of the 23 investigated datasets, which shows the practical relevance of our results. We also find examples that are not in line with our theoretical prediction, which is likely due to additional correlations (e.g. degree-degree correlations) that can invalidate theoretical predictions derived from degree-based ensembles. We finally investigate the potential impact of the self-loop paradox on the accuracy of a Graph Convolutional Network in a node classification task. In a first step, we tested the influence of self-loops on accuracy, finding that self-loops are useful for node classification in 15 of the 23 graphs (details in Appendix C). Focusing on those 15 graphs, in Fig. 2, we investigate the performance of GCNs with and without self-loops for different numbers of layers. The results show that the parity of the number of layers influences the performance of GCNs. In line with our analytical results, we find that the accuracy increase of a two-layer compared to a one-layer GCN is larger without self-loops than with self-loops for 11 of the 15 graphs. We suspect that this is due to the self-loop paradox, which states that the inclusion of self-loops limits the backflow of information from a node to itself for two layers.

## 4 Conclusion

Many GNNs add self-loops to the input graph to include feature information about a node itself at each layer. We prove that this inclusion of self-loops has a counter-intuitive effect, namely that it can decrease the information a node retains about itself in GNNs with two layers. We analytically prove this finding for random graphs with arbitrary degree sequences. Experimental validation using a node classification task in synthetically generated graphs shows that our theoretical insights hold for synthetic graphs with "noisy" topologies.

Our work shows how analytical approaches for the study of statistical ensembles of graphs with given degree sequences –which are frequently used in network science and complex systems theory– can be applied in the context of GNNs. In future work, we plan to generalise our analytical proof of the self-loop paradox for arbitrary numbers of layers $k$. We will further investigate how this effect is influenced by other properties of the graph topology, e.g. the average degree, the clustering coefficient, or degree assortativity. Another promising research direction is to improve our understanding of how the self-loop paradox may influence open challenges in GNNs like over-squashing.

## Acknowledgements

Moritz Lampert acknowledges funding from the German Federal Ministry of Education and Research, Grant No. 100582863 (TissueNet). Ingo Scholtes acknowledges funding through the Swiss National Science Foundation, Grant No. 176938.

## References

[1] Uri Alon and Eran Yahav. "On the Bottleneck of Graph Neural Networks and Its Practical Implications". In: *ICLR*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=i80OPhOCVH2. 1

[2] Rongqin Chen et al. "Redundancy-Free Message Passing for Graph Neural Networks". In: *NeurIPS*. 2022. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/1bd6f17639876b4856026744932ec76f-Abstract-Conference.html. 1, 2, 11

[3] Matthias Fey and Jan Eric Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". In: *CoRR* abs/1903.02428 (2019). URL: https://www.pyg.org/ (visited on 12/05/2022). 4, 7

[4] Justin Gilmer et al. "Neural Message Passing for Quantum Chemistry". In: *ICML*. Vol. 70. PMLR, 2017, pp. 1263–1272. URL: http://proceedings.mlr.press/v70/gilmer17a.html. 2, 11

[5] Francesco Di Giovanni et al. "On Over-Squashing in Message Passing Neural Networks: The Impact of Width, Depth, and Topology". In: *ICML*. Vol. 202. PMLR, 2023, pp. 7865–7885. URL: https://proceedings.mlr.press/v202/di-giovanni23a.html. 1

[6] William L. Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *NIPS*. 2017, pp. 1024–1034. URL: https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html. 2, 11

[7] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. "Stochastic Blockmodels: First Steps". In: *Social Networks* 5.2 (1983), pp. 109–137. DOI: 10.1016/0378-8733(83)90021-7. 2, 3

[8] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *ICLR (Poster)*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl. 1, 2, 11

[9] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. "Predict Then Propagate: Graph Neural Networks Meet Personalized PageRank". In: *ICLR (Poster)*. OpenReview.net, 2019. DOI: 10.48550/arXiv.1810.05997. 11

[10] E. A. Leicht, Petter Holme, and M. E. J. Newman. "Vertex Similarity in Networks". In: *Phys. Rev. E* 73.2 (Feb. 17, 2006), p. 026120. DOI: 10.1103/PhysRevE.73.026120. 1

[11] Michael Molloy and Bruce Reed. "A Critical Point for Random Graphs with a given Degree Sequence". In: *Random Structures & Algorithms* 6.2-3 (1995), pp. 161–180. DOI: 10.1002/rsa.3240060204. 2, 3

[12] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. "Random Graphs with Arbitrary Degree Distributions and Their Applications". In: *Phys. Rev. E* 64.2 (2001), p. 026118. DOI: 10.1103/PhysRevE.64.026118. (Visited on 06/26/2023). 1, 2

[13] Mark E. J. Newman. *Networks*. 2nd Edition. Oxford University Press, 2018. DOI: 10.1093/oso/9780198805090.001.0001. 3

[14] Jake Topping et al. "Understanding Over-Squashing and Bottlenecks on Graphs via Curvature". In: *ICLR*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=7UmjRGzp-A. 1

[15] Petar Velickovic et al. "Graph Attention Networks". In: *ICLR (Poster)*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=rJXMpikCZ. 11

[16] Herbert S Wilf. *generatingfunctionology*. CRC press, 2005. 2

[17] Keyulu Xu et al. "How Powerful Are Graph Neural Networks?" In: *ICLR*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=ryGs6iA5Km. 2, 11

[18] Keyulu Xu et al. "Representation Learning on Graphs with Jumping Knowledge Networks". In: *ICML*. Vol. 80. PMLR, 2018, pp. 5449–5458. URL: https://proceedings.mlr.press/v80/xu18c.html. 1

## A Proof of Lemma 2

*Proof.* The expected total number of incoming walks with a length of 2 is given by

$$\mathbb{E}\left(\sum_{u\in V} A_{uv}^2\right) = \langle d\rangle \cdot \langle d_N\rangle^{2-1} = \langle d\rangle \cdot \frac{\langle d^2\rangle}{\langle d\rangle} = \langle d^2\rangle. \tag{7}$$

$\tilde{\mathcal{G}}$ only differs from $\mathcal{G}$ in terms of the self-loops which increases both expected degrees by one leading to

$$\mathbb{E}\left(\sum_{u\in V} \tilde{A}_{uv}^2\right) = (\langle d\rangle + 1)\cdot(\langle d_N\rangle + 1) = \langle d^2\rangle + \langle d\rangle + \frac{\langle d^2\rangle}{\langle d\rangle} + 1. \tag{8}$$

The only closed walks that traverse two edges are those that use the same edge twice. Introducing self-loops only adds one additional option which is traversing the self-loop twice. This means the expected values are given by $\mathbb{E}\left(A_{vv}^2\right) = \langle d\rangle$ and $\mathbb{E}\left(\tilde{A}_{vv}^2\right) = \langle d\rangle + 1$. Putting everything together yields:

$$\frac{\mathbb{E}\left(A_{vv}^2\right)}{\mathbb{E}\left(\sum_{u\in V} A_{uv}^2\right)} > \frac{\mathbb{E}\left(\tilde{A}_{vv}^2\right)}{\mathbb{E}\left(\sum_{u\in V} \tilde{A}_{uv}^2\right)} \tag{9}$$

$$\Leftrightarrow \quad \frac{\langle d\rangle}{\langle d^2\rangle} > \frac{\langle d\rangle + 1}{\langle d^2\rangle + \langle d\rangle + \frac{\langle d^2\rangle}{\langle d\rangle} + 1} \tag{10}$$

$$\Leftrightarrow \quad \langle d\rangle^2 + \langle d\rangle > 0 \tag{11}$$

This statement is true since $\mathcal{G}$ and $\tilde{\mathcal{G}}$ are connected. □

The estimates used in the proof and the implications of the lemma are validated empirically in Appendix B.

## B Walk Statistics

Tab. 1 shows the average proportion of closed walks of length $k$ from a node $v$ to itself out of all walks of length $k$ ending in $v$ for $k \in \{1, 2\}$. It also includes the estimates that are used in Equation (10) of the proof. The statistics show that the proportion of closed walks of length 2 is smaller in the graph with self-loops than in the graph without self-loops which validates the theoretical findings. Furthermore, since all of the estimates are in the range of the standard deviation of the averaged proportions, it is reasonable to assume that the estimates are correct. Note that the estimates for $k = 1$ are 0 without self-loops since there is no other closed walk of length 1 than a self-loop. With self-loops, the estimate is $1/\langle d\rangle+1$ since there is one walk from or to each neighbour and the self-loop.

The statistics in Tab. 2 report the average walk proportions for larger $k$ and suggest that similar behaviour can be observed for all even $k$.

## C Real-World Examples

To investigate this effect on real-world data, we use 23 datasets from PyTorch Geometric [3]. Since we have only considered undirected graphs in our theoretical analysis, we drop the direction of the edges in the datasets that are directed. To avoid mathematical errors, we remove isolated nodes from all datasets. We further disregard existing self-loops and either remove them or add the missing ones.

Tab. 3 shows the statistics discussed in Appendix B for the real-world examples. We can see that the degrees vary far more in the real-world examples than in the synthetic graphs. Consequently, the standard deviations are larger and the estimates are less accurate. The expected pattern from our theoretical findings that the proportions are larger for walks of

| **Dataset** | ↺ | $k=1$ | $\mathbb{E}_{k=1}$ | $k=2$ | $\mathbb{E}_{k=2}$ |
|---|---|---|---|---|---|
| $p_{ii} = {}^{6}/_{100}$, $p_{ij} = {}^{3}/_{900}$ | ✔ | **0.113±0.044** | **0.102** | 0.096±0.011 | 0.093 |
| | ✘ | 0.0±0.0 | 0.0 | **0.103**±0.012 | **0.102** |
| $p_{ii} = {}^{6}/_{100}$, $p_{ij} = {}^{6}/_{900}$ | ✔ | **0.084±0.026** | **0.078** | 0.073±0.005 | 0.072 |
| | ✘ | 0.0±0.0 | 0.0 | **0.078**±0.006 | **0.078** |
| $p_{ii} = {}^{3}/_{100}$, $p_{ij} = {}^{6}/_{900}$ | ✔ | **0.116±0.045** | **0.103** | 0.096±0.011 | 0.093 |
| | ✘ | 0.0±0.0 | 0.0 | **0.104**±0.012 | **0.102** |
| $p_{ii} = {}^{1}/_{100}$, $p_{ij} = {}^{9}/_{900}$ | ✔ | **0.098±0.032** | **0.090** | 0.084±0.007 | 0.083 |
| | ✘ | 0.0±0.0 | 0.0 | **0.091**±0.008 | **0.090** |

**Table 1:** The number of closed walks with length $k$ for each node $v$ divided by the total number of walks ending in $v$ and the corresponding estimates $\mathbb{E}_{k=i}$ from Equation (10) of the proof for $k \in \{1, 2\}$. The proportions are averaged over all nodes and the standard deviation is reported. The larger value is printed in bold for each graph and walk length $k$ and their corresponding pair of proportions with and without self-loops. Additionally, the standard deviation is printed in bold if the ranges of both standard deviations do not overlap.

| **Dataset** | ↺ | $k=3$ | $k=4$ | $k=5$ | $k=6$ |
|---|---|---|---|---|---|
| $p_{ii} = {}^{6}/_{100}$, $p_{ij} = {}^{3}/_{900}$ | ✔ | **0.027±0.004** | 0.019±0.003 | **0.008±0.002** | 0.006±0.001 |
| | ✘ | 0.002±0.002 | **0.020**±0.003 | 0.002±0.001 | **0.006**±0.002 |
| $p_{ii} = {}^{6}/_{100}$, $p_{ij} = {}^{6}/_{900}$ | ✔ | **0.016±0.001** | 0.012±0.001 | **0.004±0.001** | 0.003±0.001 |
| | ✘ | 0.001±0.001 | **0.012**±0.002 | 0.001±0.001 | **0.003**±0.001 |
| $p_{ii} = {}^{3}/_{100}$, $p_{ij} = {}^{6}/_{900}$ | ✔ | **0.026±0.003** | 0.018±0.002 | **0.007±0.001** | 0.005±0.001 |
| | ✘ | 0.001±0.001 | **0.020**±0.003 | 0.001±0.001 | **0.005**±0.001 |
| $p_{ii} = {}^{1}/_{100}$, $p_{ij} = {}^{9}/_{900}$ | ✔ | **0.020±0.002** | 0.014±0.002 | **0.006±0.001** | 0.004±0.001 |
| | ✘ | 0.001±0.001 | **0.015**±0.002 | 0.001±0.000 | **0.004**±0.001 |

**Table 2:** The average proportions as in Tab. 1 for larger $k$.

length 2 without self-loops cannot be observed in all of the datasets. Although the estimates falsely predict the pattern in all datasets, they still give a good indication of when the pattern exists because the difference is very small ($< 0.001$) in most datasets where the pattern does not exist. The predicted pattern also continues for larger walk lengths for some of the datasets which further supports our hypothesis from Appendix B that this alternating pattern continues for larger walk lengths.

The aforementioned datasets are further used to evaluate the performance of the GCN model on node classification tasks. The training parameters are set to similar values as above. We train the GCNs for 70 epochs on 80% of the nodes. An Adam optimizer is used with learning rate 0.01 and we include a dropout layer with a dropout rate of 0.2 after each GCN layer.

Our theoretical results make statements about the amount of information a node retains about itself for MPNN architectures with two layers. Thus, it is reasonable to assume that we can only expect results that are in line with our theoretical findings if the node's own feature is useful for the node classification task. We measure the usefulness of a node's own feature on the performance of single-layered GCNs. The node's own feature does not influence the predictions of single-layered GCNs without self-loops and thus if that GCN performs better than the one with self-loops, the node's own feature can be assumed to be not useful or even harmful to the task. We can observe those results for 8 of the investigated datasets and report only the remaining 15 datasets in Fig. 2.

The results show that the parity of the number of message passing layers strongly influences the performance of GCNs. In line with our analytical results, we further find that for 11 out of 15 data sets the accuracy increase of a two-layer compared to a one-layer GCN is larger for an input graph without self-loops than a graph with self-loops. We suspect that this is

| **Dataset** | ↺ | $k=1$ | $\mathbb{E}_{k=1}$ | $k=2$ | $\mathbb{E}_{k=2}$ | $k=3$ | $k=4$ |
|---|---|---|---|---|---|---|---|
| Planetoid: | ✔ | **0.275±0.128** | **0.204** | 0.174±0.112 | 0.084 | **0.092**±0.106 | 0.069±0.105 |
| Cora | ✘ | 0.0±0.0 | 0.0 | **0.213**±0.207 | **0.092** | 0.017±0.032 | **0.090**±0.204 |
| Planetoid: | ✔ | **0.350±0.140** | **0.265** | 0.274±0.136 | 0.126 | **0.200**±0.157 | 0.176±0.164 |
| CiteSeer | ✘ | 0.0±0.0 | 0.0 | **0.399**±0.311 | **0.145** | 0.017±0.040 | **0.287**±0.350 |
| Planetoid: | ✔ | **0.339±0.169** | **0.182** | **0.104**±0.068 | 0.056 | **0.033**±0.037 | **0.018**±0.024 |
| PubMed | ✘ | 0.0±0.0 | 0.0 | 0.092±0.094 | **0.060** | 0.002±0.006 | 0.018±0.031 |
| CitationFull: | ✔ | **0.272±0.165** | **0.144** | 0.152±0.128 | 0.046 | **0.076**±0.120 | 0.057±0.117 |
| DBLP | ✘ | 0.0±0.0 | 0.0 | **0.186**±0.235 | **0.049** | 0.005±0.013 | **0.089**±0.233 |
| Amazon: | ✔ | **0.079**±0.096 | **0.027** | 0.022±0.045 | 0.006 | **0.006**±0.033 | 0.004±0.032 |
| Computers | ✘ | 0.0±0.0 | 0.0 | **0.023**±0.068 | **0.006** | 0.002±0.012 | **0.005**±0.058 |
| Amazon: | ✔ | **0.081**±0.096 | **0.031** | 0.026±0.045 | 0.010 | **0.008**±0.035 | 0.006±0.034 |
| Photo | ✘ | 0.0±0.0 | 0.0 | **0.028**±0.072 | **0.010** | 0.004±0.012 | **0.007**±0.065 |
| Coauthor: | ✔ | **0.170±0.119** | **0.101** | 0.090±0.064 | 0.052 | **0.030**±0.036 | **0.014**±0.022 |
| CS | ✘ | 0.0±0.0 | 0.0 | **0.092**±0.077 | **0.055** | 0.011±0.015 | 0.012±0.021 |
| Wikipedia: | ✔ | **0.115**±0.118 | **0.035** | 0.028±0.049 | 0.009 | **0.010**±0.024 | **0.006**±0.013 |
| Chameleon | ✘ | 0.0±0.0 | 0.0 | **0.029**±0.055 | **0.009** | 0.005±0.012 | 0.006±0.012 |
| Wikipedia: | ✔ | **0.091**±0.107 | **0.013** | **0.020**±0.036 | 0.002 | **0.003**±0.010 | **0.001**±0.004 |
| Squirrel | ✘ | 0.0±0.0 | 0.0 | 0.018±0.036 | **0.002** | 0.001±0.004 | 0.001±0.004 |
| Actor | ✔ | **0.248±0.159** | **0.125** | **0.081**±0.081 | 0.019 | **0.015**±0.032 | **0.005**±0.013 |
| | ✘ | 0.0±0.0 | 0.0 | 0.077±0.090 | **0.019** | 0.001±0.005 | 0.003±0.010 |
| Airports: | ✔ | **0.196±0.173** | **0.042** | **0.030**±0.050 | 0.010 | **0.007**±0.031 | 0.004±0.029 |
| USA | ✘ | 0.0±0.0 | 0.0 | 0.027±0.071 | **0.011** | 0.003±0.006 | **0.005**±0.058 |
| Airports: | ✔ | **0.113**±0.126 | **0.032** | **0.020**±0.023 | 0.014 | **0.004**±0.003 | **0.003**±0.003 |
| Europe | ✘ | 0.0±0.0 | 0.0 | 0.017±0.019 | **0.014** | 0.003±0.002 | 0.003±0.003 |
| Airports: | ✔ | **0.160±0.151** | **0.061** | **0.033**±0.012 | 0.029 | **0.010**±0.007 | **0.009**±0.007 |
| Brazil | ✘ | 0.0±0.0 | 0.0 | 0.029±0.012 | **0.030** | 0.008±0.007 | 0.008±0.007 |
| Twitch: | ✔ | **0.107**±0.120 | **0.030** | **0.011**±0.027 | 0.004 | **0.000**±0.003 | **0.000**±0.000 |
| DE | ✘ | 0.0±0.0 | 0.0 | 0.009±0.025 | **0.004** | 0.000±0.000 | 0.000±0.000 |
| Twitch: | ✔ | **0.217±0.156** | **0.092** | **0.051**±0.068 | 0.017 | **0.007**±0.021 | **0.002**±0.006 |
| EN | ✘ | 0.0±0.0 | 0.0 | 0.047±0.073 | **0.017** | 0.001±0.003 | 0.001±0.004 |
| Twitch: | ✔ | **0.163±0.144** | **0.056** | **0.025**±0.046 | 0.008 | **0.002**±0.009 | **0.001**±0.003 |
| RU | ✘ | 0.0±0.0 | 0.0 | 0.022±0.044 | **0.008** | 0.000±0.001 | 0.001±0.002 |
| WebKB: | ✔ | **0.350±0.141** | **0.248** | 0.132±0.119 | 0.049 | **0.046**±0.053 | **0.025**±0.024 |
| Cornell | ✘ | 0.0±0.0 | 0.0 | **0.143**±0.150 | **0.052** | 0.006±0.012 | 0.023±0.031 |
| WebKB: | ✔ | **0.353±0.133** | **0.247** | 0.118±0.114 | 0.042 | **0.035**±0.042 | **0.018**±0.018 |
| Texas | ✘ | 0.0±0.0 | 0.0 | **0.121**±0.142 | **0.043** | 0.003±0.006 | 0.016±0.022 |
| WebKB: | ✔ | **0.310±0.133** | **0.218** | 0.119±0.106 | 0.045 | **0.036**±0.044 | **0.018**±0.019 |
| Wisconsin | ✘ | 0.0±0.0 | 0.0 | **0.123**±0.129 | **0.047** | 0.004±0.008 | 0.016±0.019 |
| LastFM- | ✔ | **0.251±0.163** | **0.121** | **0.098**±0.086 | 0.038 | **0.026**±0.041 | **0.011**±0.022 |
| Asia | ✘ | 0.0±0.0 | 0.0 | 0.094±0.101 | **0.039** | 0.004±0.008 | 0.009±0.021 |
| Amazon- | ✔ | **0.137±0.039** | **0.116** | 0.094±0.035 | 0.075 | **0.057**±0.034 | **0.050**±0.032 |
| Ratings | ✘ | 0.0±0.0 | 0.0 | **0.101**±0.044 | **0.081** | 0.042±0.032 | 0.050±0.032 |
| Roman- | ✔ | **0.271±0.059** | **0.256** | 0.246±0.030 | 0.234 | **0.174±0.030** | 0.149±0.030 |
| Empire | ✘ | 0.0±0.0 | 0.0 | **0.316**±0.059 | **0.306** | 0.059±0.043 | **0.172**±0.043 |
| Mine- | ✔ | **0.113±0.011** | **0.113** | 0.113±0.005 | 0.112 | **0.068±0.004** | **0.056**±0.003 |
| sweeper | ✘ | 0.0±0.0 | 0.0 | **0.127±0.006** | **0.126** | 0.048±0.003 | 0.054±0.003 |

**Table 3:** The average proportions and statistics as in Tab. 1 and Tab. 2 for real-world datasets. The names of the datasets that are in line with our theoretical results are marked in orange. As in Tab. 1, the larger number of each value pair is printed in bold. Note that some values are rounded to the same values. In this case, the value that was larger before rounding is printed in bold.

due to the self-loop paradox, which states that the inclusion of self-loops actually limits the backflow of information from a node to itself for two message passing layers.

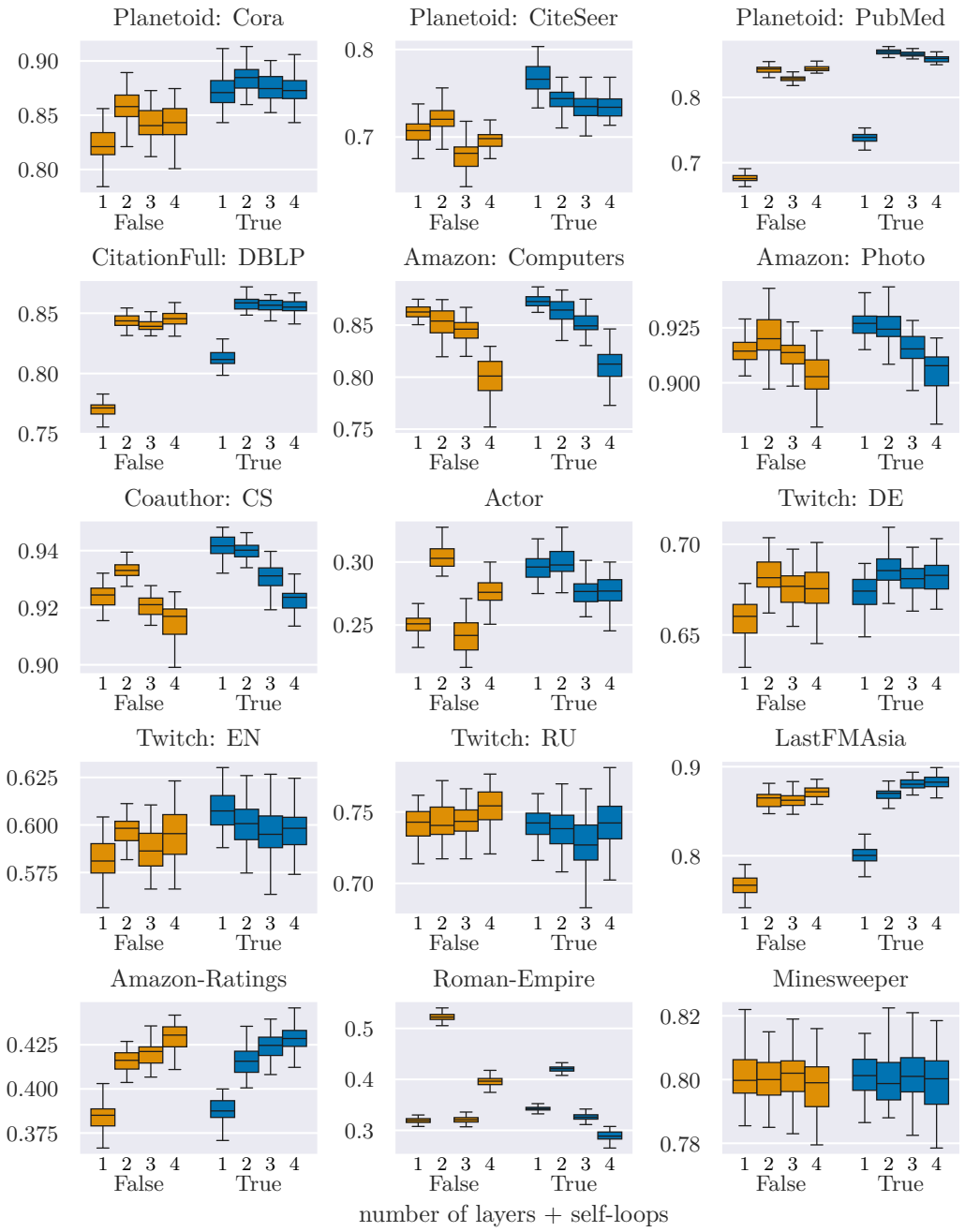


**Figure 2:** Test set accuracies as in Fig. 1 for real-world examples. Outliers outside of the whiskers are omitted for readability.

## D The Self-Loop Paradox for Other GNNs

So far, we only considered the Graph Convolutional Network (GCN) [8] although there exists a large variety of other GNN architectures. Lemma 1 holds for all Message Passing GNNs (MPNNs) –sometimes also called Weisfeiler-Leman (WL) test based GNNs [2]. By definition of Gilmer et al. [4], this includes all GNNs that can be expressed with a message function

$$\mathbf{m}_v^{(k+1)} = \sum_{u \in N(v)} M^{(k)} \left( \mathbf{h}_v^{(k)}, \mathbf{h}_u^{(k)}, \mathbf{e}_{vu} \right) \tag{12}$$

and an update function

$$\mathbf{h}_v^{(k+1)} = U^{(k)} \left( \mathbf{h}_v^{(k)}, \mathbf{m}_v^{(k+1)} \right). \tag{13}$$

This includes GNNs like the GCN [8], GraphSAGE [6] or the Graph Isomorphism Network (GIN) [17] but does not include models like Personalized Propagation of Neural Predictions (PPNP) [9].

In contrast to the original formulation of the GCN, this general framework does not rely on self-loops for all nodes since it includes the node's own feature in the update step. Our theoretical findings, on the other hand, are based on the assumption that the node's own feature is included via self-loops. This means our findings do not apply to all MPNNs but only to those that can be reformulated using a modified message step that includes self-loops

$$\mathbf{m}_v^{(k+1)} = \sum_{u \in N(v) \cup \{v\}} M^{(k)} \left( \mathbf{h}_v^{(k)}, \mathbf{h}_u^{(k)}, \mathbf{e}_{vu} \right) \tag{14}$$

and an update step that excludes the node's own feature

$$\mathbf{h}_v^{(k+1)} = U^{(k)} \left( \mathbf{m}_v^{(k+1)} \right). \tag{15}$$

This includes the GCN as defined in Equation (4) but also the Graph Attention Network (GAT) by Velickovic et al. [15]:

$$\mathbf{m}_v^{(k+1)} = \sum_{u \in N(v) \cup \{v\}} a_{vu}^{(k)} \mathbf{h}_u^{(k)} W^{(k)} \tag{16}$$

$$\mathbf{h}_v^{(k+1)} = \sigma \left( \mathbf{m}_v^{(k+1)} + \mathbf{b}^{(k)} \right) \tag{17}$$

Technically, this is only true if one disregards that the Softmax function considers the whole neighbourhood $N(v)$ to calculate the attention weights $a_{vu}$ which would require a message function $M^{(k)} \left( \mathbf{h}_v^{(k)}, \mathbf{h}_u^{(k)}, \mathbf{e}_{vu}, \{h_w^{(k)} : w \in N(v)\} \right)$. Other models like GraphSAGE and GIN do not fall into this category since they both either concatenate or add the node's feature $\mathbf{h}_v^{(k)}$ to the aggregated message $\mathbf{m}_v^{(k+1)}$ in the update function.