EFFECTIVE DIAMETER: OVER-SMOOTHING IN THE CONTEXT OF LIMITED PROBLEM RADII

Anonymous authors
Paper under double-blind review

ABSTRACT

Over-smoothing is widely considered to be one of the main reasons why Graph Neural Networks (GNNs) achieve their best performance with only a few message passing layers. In this paper, we critically examine this assumption on real-world literature benchmarks. Over-smoothing is theoretically well understood and effectively mitigated by established techniques. In experiments on common node classification benchmarks, we show that these mitigation techniques largely stabilise GNN performance for depths of up to 64 layers, but deep models still fail to significantly outperform shallow ones. As an alternative explanation for shallow optimal GNN models, we propose the *effective diameter*, a data-based statistic free of any model assumptions that estimates the size of the task-relevant receptive field. It builds on the shell *separability gap*, which aims to measure the feature-based class discriminative power of a k-shell. On all investigated datasets, the separability gap decays within a few hops, and the best-performing models rarely lie beyond the estimated effective diameter. This suggests that the small optimal depth of GNNs is caused by the lack of task-relevant information in large receptive fields rather than by over-smoothing.

1 INTRODUCTION

A graph is a versatile data structure that is well-suited for describing complex systems, offering a mathematical foundation to analyse real-world systems using Graph Neural Networks (GNNs). Gilmer et al. (2017) introduced the Message Passing Neural Network (MPNN) framework, which serves as the foundation for most modern GNNs, with powerful variants like Graph Convolutional Networks (GCNs) (Kipf & Welling, 2017) and Graph Attention Networks (GATs) (Veličković et al., 2018; Brody et al., 2022) achieving state-of-the-art results on many graph learning tasks. These advances have spurred a wide range of applications. For example, GNNs are now routinely applied in chemistry and drug discovery for molecular property prediction (Li et al., 2021), in large-scale recommendation engines in social networks (Ying et al., 2018; Zhao et al., 2025; Liu et al., 2024), and in spatial–temporal forecasting for traffic prediction like for Google Maps (Yu et al., 2017; Derrow-Pinion et al., 2021). Theoretical work on GNNs has explored several challenges that limit their capabilities for graph learning tasks. Most prominently, the problems of over-smoothing, over-squashing, expressivity, and robustness are discussed in many publications over the past eight years (Rusch et al., 2023a; Akansha, 2025; Zhang et al., 2025; Morris et al., 2019; Zhang et al., 2024; Günnemann, 2022). While addressing all four challenges remains an active focus of current GNN research, this paper critically examines over-smoothing.

Over-smoothing refers to the phenomenon wherein node representations converge to a single representation with an increasing number of message passing steps and hence become uninformative. In many publications, this is assumed to be the main reason for a relatively small optimal number of message passing layers compared to other neural architectures, like state-of-the-art Convolutional Neural Networks (CNNs) with hundreds of layers (He et al., 2016; Tan & Le, 2021; Howard et al., 2019). In this paper, we critically investigate the impact of over-smoothing in real-world datasets. Based on the extensive body of literature (Section 3) and experimental results on over-smoothing, we call into question the importance of working with arbitrary large receptive fields. We build on the analysis of over-smoothing and propose a method to measure the relevant problem radius. Our main contributions are summarised as follows: We extensively analyse the effect of over-smoothing on GNN performance in real-world literature benchmark datasets. We find that over-smoothing can

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

be successfully mitigated by mitigation techniques, but deep GNNs nevertheless fail to significantly outperform shallow ones. This indicates a negligible impact of over-smoothing on the investigated datasets (Section 4). Further, to quantify the relevance of different-sized receptive fields, we propose to measure the *shell separability gap*, which is the relative difference of the feature distance between shell nodes with different central node classes and the feature distance between shell nodes within central node classes (Section 5). We introduce a new data-based metric, the *effective diameter* based on the shell separability gap, to estimate the size of the task-relevant receptive field (Section 5).

2 PRELIMINARIES

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with node set $\mathcal{V}$ and edge set $\mathcal{E}$. The graph $\mathcal{G}$ can be represented via an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{\mathcal{V}\times\mathcal{V}}$ with $a_{vu} \neq 0$ if and only if $(v, u) \in \mathcal{E}$. Each node $v \in \mathcal{V}$ has a neighbourhood $\mathcal{N}(v) = \{u : (v, u) \in \mathcal{E}\}$ of degree $d(v) = |\mathcal{N}(v)|$, a k-hop neighbourhood $\mathcal{N}^{(:k)}(v) = \{u \in \mathcal{V} \mid \delta(v, u) \leq k\}$, where $\delta(v, u)$ denotes the shortest path distance between v and u in the graph $\mathcal{G}$, and features $\boldsymbol{h}_v^{(0)} \in \mathbb{R}^{d_0}$, which are the rows of a matrix $\boldsymbol{H}^{(0)} \in \mathbb{R}^{\mathcal{V}\times d_0}$. Graph Neural Networks (GNNs) are neural networks designed to process graph-structured data. Most GNNs fall into the class of MPNNs, in which layers of message passing functions $M(\cdot)$ and update functions $U(\cdot)$ are iteratively composed. In layer ℓ, a new hidden node representation is computed as

$$\boldsymbol{h}_v^{(\ell+1)} = U\left(\boldsymbol{h}_v^{(\ell)}, M\left(\boldsymbol{h}_v^{(\ell)}, \{\boldsymbol{h}_u^{(\ell)} : u \in \mathcal{N}(v)\}\right)\right).$$

For an L-layer MPNN, the receptive field of a node v refers to the set of nodes whose initial features $\boldsymbol{h}^{(0)}$ influence the final embedding $\boldsymbol{h}_v^{(L)}$, i.e., all nodes with $\delta(v, u) \leq L$.

3 RELATED WORK

We review the existing theory on over-smoothing and approaches to measure and to mitigate it.

3.1 OVER-SMOOTHING

Kipf & Welling (2017, Appendix B) first discovered a performance peak of GNNs, here the Graph Convolution Network (GCN), at around 7–8 message passing layers. Li et al. (2018b) further investigated this phenomenon based on the simplification of graph convolutions as a special form of Laplacian smoothing and showed that infinitely repeated applications of the message passing operator lead node embeddings to converge to a constant vector. Further, they demonstrated empirically for a small graph dataset (Zachary, 1977) that repeated graph convolutions first improve the class separation in the latent space before collapsing the representations.

Chen et al. (2020a) approached the phenomenon empirically and were among the first trying to measure over-smoothing. They propose to use the cosine-based Mean Average Distance (MAD) and the MADGap as a quantitative metric for the smoothness and over-smoothness of the graph, respectively. Their experiments suggest a connection between over-smoothness, performance, and the information-to-noise ratio, quantified by the proportion of same-class nodes in an L-hop neighbourhood, where L corresponds to the number of message passing layers.

Oono & Suzuki (2020) investigated the asymptotic behaviour of the expressive power of GCNs as the number of layers tends to infinity. They show that the node representations converge to a single signal determined by the node's connected component and degree, as well as the exponential nature of this asymptotic behaviour. They show that $d_{\mathcal{M}}(\boldsymbol{h}_v^{(\ell)}) \leq s\,\lambda\, d_{\mathcal{M}}(\boldsymbol{h}_v^{(\ell-1)})$, where $d_{\mathcal{M}}(\boldsymbol{h}_v^{(\ell)})$ denotes the distance of a node representation $\boldsymbol{h}_v$ in layer ℓ to the common representation in subspace $\mathcal{M}$ of the same connected component and node degree. This measures the smoothness of the graph. The over-smoothing factor $s\lambda$ comprises λ, the non-trivial eigenvalue of the propagation matrix with the largest absolute value, and s, the largest singular value of all weight matrices $\boldsymbol{W}^{(\ell)}$. They argue that for most graphs, $s\lambda < 1$, implying inevitable over-smoothing.

Keriven (2022) provided a theoretical analysis of the effect of message passing, focusing not just on the asymptotic behaviour but also the optimal GNN depth. Using a stochastic latent-space random-graph model and analysing the mean-aggregation dynamics via the ergodic theorem, he showed

that a limited number of message passing layers improves learning due to faster shrinking of non-principal than of principal components and compression of inter-community variance before complete collapse occurs.

Building on the curvature-based framework for over-squashing introduced by Topping et al. (2022), Nguyen et al. (2023) connected positive edge curvature to over-smoothing. They showed that higher curvature κ implies higher neighbourhood overlap between adjacent nodes

$$\frac{|\mathcal{N}(v) \cap \mathcal{N}(u)|}{\max(d(v), d(u))} \geq \kappa(v, u),$$

and hence contributes to the smoothing of node features.

Keriven (2026) examined over-smoothing from an optimisation perspective. He introduced the idea of *backward over-smoothing* and argues that "as soon as the last layer of the GNN is 'trained', then gradients vanish at every layer and the GNN cannot train anymore", leading to stationary points.

Arnaiz-Rodriguez & Errica (2026) challenge common beliefs about over-smoothing and over-squashing by constructing theoretically sound counterexamples. They question the performance-degrading effect of over-smoothing. They instead emphasise the importance of node separability, which can be influenced by many effects, and suggest further research in this direction.

3.2 Measuring over-smoothing

Several measures for over-smoothing have been proposed in recent years, which differ primarily in two design choices: (1) the metrics used to quantify the distance between node representations, and (2) the pairs of nodes between which this distance is measured. The most prominent measure is the *Dirichlet energy*, which is based on squared differences between representations of neighbouring nodes (Rusch et al., 2023a; Cai & Wang, 2020). Di Giovanni et al. (2023) normalise the Dirichlet energy to eliminate the feature scale and isolate the frequency composition of the features. Wu et al. (2023) propose a *node similarity* measure over all node pairs, also based on squared differences. The *Mean Average Distance* (MAD) (Chen et al., 2020a) applies the cosine distance between each node and its neighbours within a given k-hop neighbourhood. Zhang et al. (2026) propose to measure over-smoothing using the *numerical rank* (NumRank) and the *effective rank* (Erank) of the whole feature matrix. We provide further details on these measures in Appendix A. To measure over-smoothing in a GNN, the appropriate distance metric must be selected depending on the message passing scheme. The degree-normalised version of the Dirichlet energy (Cai & Wang, 2020) is expected to converge to zero for GCNs, as they collapse node representations to a constant vector scaled by the degree (Oono & Suzuki, 2020; Cai & Wang, 2020). The Dirichlet energy as used by Rusch et al. (2023a) and the node similarity (Wu et al., 2023) converge to zero if and only if all node representations become identical. This over-smoothing behaviour is expected for models with row-stochastic message passing operators like GATs. MAD and the rank-based measures can be applied to GCNs and GATs.

3.3 Methodologies countering over-smoothing

Several architectural interventions have been proposed to counter over-smoothing. Zhao & Akoglu (2020) introduced PairNorm to reduce the effect of over-smoothing by stabilising the sum of the distances between node pairs with and without edges. Chen et al. (2020b) showed that simple skip connections can enable stable performance of deep GCNs. Specifically, they introduced GCNII, which adds the initial embedding to each hidden embedding by a proportion α and an identity mapping to each weight matrix $\boldsymbol{W}^{(\ell)}$ by a proportion β, where α and β are hyperparameters. Scholkemper et al. (2025) proved theoretically that a GCN incorporating both batch-normalisation and residual connections does not suffer from over-smoothing. These theoretical findings are supported by empirical results on GNNs with the most common message passing methods. Rusch et al. (2023b) further developed the idea of residual layers to gradient gating (G^2) by replacing the fixed proportion β with learnable element-wise rates $\boldsymbol{\tau}^{(\ell)}$. In the G^2 model, $\boldsymbol{\tau}^{(\ell)}$ is given by a learnable function of the hidden features and the graph structure. Rusch et al. (2023a) surveyed these and other mitigation techniques and categorised them into (1) normalisation and regularisation, (2) change of the GNN dynamics, and (3) architectural enhancement through residual connections.

The extensive work on over-smoothing has led to a good theoretical understanding of this phenomenon. However, a lot of these works focus on proving asymptotic behaviour. Our work tries to put these theoretical results in the context of extensive experiments on real-world datasets.

4 Empirical analysis of over-smoothing and GNN performance

Over-smoothing is often held accountable for the limited performance of GNNs. The reasoning behind this conclusion is that over-smoothing prevents building deep GNNs as repeated message passing makes nodes indistinguishable. This reasoning has the underlying assumption that more message passing improves performance. We tested this assumption empirically. Our experimental results show that mitigation techniques mostly prevent over-smoothing, but deep GNNs still fail to significantly outperform shallow models on the analysed real-world datasets.

4.1 Experiments: Over-smoothing in real-world data

We conducted experiments on common node-classification datasets using standard GNNs, i.e., GCN Kipf & Welling (2017), GATv2 Brody et al. (2022), and GraphSAGE Hamilton et al. (2018), and three methods designed to mitigate over-smoothing, namely PairNorm (Zhao & Akoglu, 2020), GCNII (Chen et al., 2020b), and G^2 (Rusch et al., 2023b). We combined PairNorm and G^2 with the three standard GNN backbones. For each model–dataset combination, we performed hyperparameter optimisation and training separately for different model depths with an exponentially increasing number of message passing layers, ranging from 2 to 64. We then evaluated both the classification performance (Table 1) and relevant over-smoothing measures (Figure 1) of these optimised models per model depth. Appendix A and B provide more details about the over-smoothing measures and the experimental set-up. We restrict our analysis of over-smoothing to node classification.

The results show clearly that over-smoothing is of negligible relevance in the examined datasets. For the GCN with an increasing number of message passing layers, we can observe a performance decrease in Table 1 and often as well a decreasing MAD in Figure 1. A concurrency of these effects, like on Cora in the 4th layer, could indicate over-smoothing. However, on other datasets, like PubMed and Photo, the performance drops in models with few layers, where the MAD is still stable. In these cases, the performance drop can therefore not stem from over-smoothing as quantified via the MAD. In contrast, the GCNII and the G^2-GCN, which are theoretically capable of compensating for over-smoothing, exhibit none of these behaviours, maintaining relatively stable accuracy and MAD across layers. PairNorm at least reduces performance degradation and smoothing. However, in most cases, the optimal performance is achieved with a shallow model. In those cases where the best accuracy is obtained with a large number of layers, like for the GCNII on PubMed and Photo, the improvement is marginal and does not justify the increasing computational cost. These findings are consistent across additional experiments with GATv2 and GraphSAGE (Appendix D).

4.2 Findings: Shallow optimal models

We can conclude from our experimental results in Section 4.1 that although mitigating over-smoothing works and stabilises model performance with respect to model depth, it does not necessarily lead to superior performance of deep models compared to shallow models. This suggests that over-smoothing is a less relevant problem in real-world datasets, that are currently established in the Graph Representation Learning literature, than often assumed. The superior performance of mitigation methods with small depth in some datasets, like Cora or Roman-Empire, cannot be explained by preventing over-smoothing. Skip connections and normalisations are known to generally improve the training behaviour of neural networks (He et al., 2016; Li et al., 2018a; Ioffe & Szegedy, 2015; Santurkar et al., 2018; Luo et al., 2024). Further, the over-smoothing measures in Figure 1 show that over-smoothing is not an issue at depths of 2–4 layers. The effect of mitigating over-smoothing can be observed by comparing the impact of model depth on the model performance. The fact that increased depth does not lead to superior performance, even though over-smoothing is prevented, shows that over-smoothing does not limit GNN performance on the datasets studied in Section 4.1. This finding raises the question of why deep GNNs fail to outperform shallow ones if over-smoothing is not the reason. We now investigate the size of the relevant receptive field as an alternative explanation for small optimal model depths.

Table 1: Comparison of a GCN and three methods to mitigate over-smoothing for a range of model depths. The accuracy is given in % with the standard deviation over 10 random parameter initialisations in brackets. The best-performing model depth is highlighted in bold. For two settings, our hardware ran out of memory (OOM).

Dataset	Model	Layers					
		2	4	8	16	32	64
Cora	GCN	**74.1 (2.4)**	36.1 (13.4)	31.9 (0.0)	31.9 (0.0)	31.9 (0.0)	31.9 (0.0)
	GCN + Pairnorm	**66.2 (6.4)**	61.7 (6.3)	59.6 (3.7)	38.5 (7.9)	33.0 (3.4)	34.2 (4.9)
	GCNII	85.0 (0.3)	84.9 (2.0)	84.4 (2.4)	84.6 (3.0)	84.6 (3.5)	**86.8 (0.6)**
	G^2	70.5 (6.1)	74.7 (0.5)	73.1 (2.5)	74.7 (0.7)	**74.9 (0.6)**	70.7 (5.9)
CiteSeer	GCN	**75.5 (0.8)**	74.4 (0.4)	73.6 (0.7)	65.6 (8.0)	60.7 (5.3)	18.1 (0.0)
	GCN + Pairnorm	**68.3 (0.9)**	67.4 (1.5)	66.6 (4.2)	65.0 (5.5)	44.5 (7.2)	29.2 (6.6)
	GCNII	77.5 (0.6)	77.5 (0.6)	**78.1 (0.6)**	77.7 (0.5)	77.4 (1.1)	77.9 (0.6)
	G^2	75.0 (3.7)	74.4 (5.1)	**76.3 (0.6)**	73.6 (5.3)	75.8 (0.4)	73.9 (5.3)
PubMed	GCN	**87.3 (0.4)**	85.8 (0.2)	83.9 (3.5)	80.4 (6.0)	55.9 (14.6)	40.7 (0.2)
	GCN + Pairnorm	**88.2 (0.4)**	86.5 (0.3)	85.7 (0.4)	84.5 (0.3)	84.5 (0.5)	76.5 (5.2)
	GCNII	88.0 (0.5)	87.9 (0.4)	87.8 (0.4)	87.8 (0.5)	88.0 (0.4)	**88.6 (0.4)**
	G^2	**88.2 (0.3)**	86.9 (1.4)	86.2 (2.7)	87.4 (0.6)	84.4 (6.6)	82.2 (14.4)
Roman-Empire	GCN	**35.7 (8.7)**	22.1 (8.5)	15.0 (2.0)	14.5 (1.2)	13.9 (0.1)	14.0 (0.3)
	GCN + Pairnorm	**17.8 (2.7)**	16.6 (1.5)	16.3 (1.5)	15.5 (2.6)	14.7 (1.1)	13.9 (0.5)
	GCNII	55.1 (10.2)	**57.7 (13.7)**	43.4 (13.7)	35.5 (12.9)	38.6 (15.2)	38.7 (13.8)
	G^2	52.2 (14.5)	54.3 (12.8)	52.8 (13.5)	**54.9 (12.7)**	54.4 (11.7)	OOM
Photo	GCN	**87.1 (0.9)**	85.1 (1.4)	71.7 (14.8)	49.6 (20.3)	27.5 (18.7)	23.0 (16.8)
	GCN + Pairnorm	**88.0 (1.8)**	82.9 (2.4)	84.0 (2.9)	84.5 (2.6)	83.5 (1.9)	74.2 (9.1)
	GCNII	89.1 (1.8)	88.6 (2.1)	88.8 (1.9)	**89.3 (1.5)**	88.9 (1.6)	88.7 (1.2)
	G^2	71.6 (15.0)	83.2 (6.8)	86.4 (3.4)	82.6 (8.2)	**86.9 (2.6)**	74.8 (10.5)
Computers	GCN	**76.9 (4.0)**	71.4 (10.6)	36.9 (21.2)	28.6 (16.3)	32.2 (13.5)	15.6 (16.3)
	GCN + Pairnorm	**81.8 (2.5)**	77.9 (3.1)	78.5 (3.0)	75.3 (4.1)	47.2 (24.6)	50.0 (16.7)
	GCNII	**82.4 (1.6)**	80.5 (2.1)	80.3 (2.2)	80.3 (2.0)	81.0 (1.8)	81.2 (1.9)
	G^2	70.7 (2.9)	71.6 (3.2)	**73.7 (2.0)**	70.9 (3.7)	72.8 (7.6)	OOM

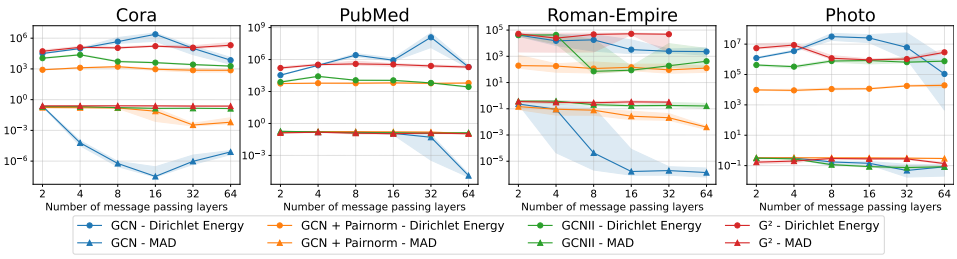

Figure 1: Median over-smoothing measures over the whole dataset for tested GCN variants (further results in Appendix A) at different depths. The error bands represent the 25- and 75-percentiles over 10 random parameter initialisations.

5 Effective Diameter

We introduce the *effective diameter* as a statistic to estimate the size of the task-relevant receptive field. For many learning tasks, a small receptive field might be sufficient to obtain the relevant information. If the problem radius is smaller than or similar to the number of message passing steps to achieve optimal smoothing, as described by Keriven (2022), the performance will not suffer from over-smoothing and adding more message passings is not necessary to achieve the best performance.

Let $\mathrm{p}(k)$ be the best performance of an optimal Graph Neural Network with depth $L = k$, i.e., with a receptive field of $\mathcal{N}^{(:k)}(v)$. We then define the *effective diameter* of a graph $\mathcal{G}$ as the depth of the overall best-performing model $\phi_{\text{eff}}(\mathcal{G}) = \arg\max_k \mathrm{p}(k)$. The optimal message passing model therefore needs at most $\phi_{\text{eff}}(\mathcal{G})$ message passing steps to observe all task-relevant information. As the optimal model architecture and training protocol are not known in general, the exact $\phi_{\text{eff}}(\mathcal{G})$ cannot be determined, only estimated. A heuristic estimation via model performance is expensive as it would require testing different model architectures with an extensive grid search for hyperparameter tuning. Therefore, we propose a data-based estimator $\hat{\phi}_{\text{eff}}(\cdot)$ for the effective diameter, which is free of any model assumptions.

Our measure $\hat{\phi}_{\text{eff}}(\cdot)$ is a conservative estimator of $\phi_{\text{eff}}(\cdot)$, which indicates the maximal depth of a useful receptive field. If, at a certain number of message passings k, the additional information that is aggregated looks the same for all classes, then it is unlikely to be useful for node classification. We consider the k-shell of a node, that is, $\mathcal{N}_v^{(k)} = \{u \in \mathcal{V} | \delta(v,u) = k\}$ the set of nodes at exactly distance k from the considered node, and ask whether the information found in that shell differs between classes. If it does not, we expect a model to not improve by aggregating that shell. We estimate $\phi_{\text{eff}}(\mathcal{G})$ by sampling shortest paths in $\mathcal{G}$ and measuring the class-conditioned distance of the node features per shell in these paths. The path sampling process is described in Section 5.1, followed by the definition of the class-conditioned feature distance in Section 5.2. We report and interpret the results in Section 5.3 and discuss limitations in Section 5.4.

5.1 Path Sampling

We sample propagation paths in the graph to collect nodes of different k-shells for our estimator. This approach allows for efficient computation while inducing a structural bias of the graph in our estimator. A sample path of length ℓ is a node sequence $p = (v_0, \dots, v_\ell)$ with $(v_m, v_{m+1}) \in \mathcal{E}$ and $\delta(v_0, v_m) = m$ for all $m \in [0, \dots, \ell - 1]$, so that every prefix of p is itself a shortest path. We call v_0 the source of p. The node v_m at position m lies in the shell m of v_0. If we apply a GNN on the graph, the features of v_m are aggregated at v_0 after m message passing steps. Further, we define that every path inherits the class of its source, $c(p) := c_{v_0}$. Nodes in shell m are considered for the class of the path source for which they should be informative. The individual class of nodes in shell m is only considered in the context of paths for which they are the source. This set-up mirrors the node-classification task and captures what the neighbourhood of a node looks like given the class of that node. By comparing shells within these sample paths, we obtain node samples that are dependent across shells and allow for a consistent comparison.

We formalise the sampling process of propagation paths in Algorithm 1. The path sampling starts by randomly choosing source nodes $\mathcal{V}_{\text{src}} \subset \mathcal{V}$. We stratify this source node selection to balance the classes, i.e., $|\mathcal{V}_{\text{src}}^c| = |\mathcal{V}_{\text{src}}^{c'}|$ for all $c, c' \in \mathcal{C}$, so that every class contributes the same share of the sampled paths. Per source node, we obtain the shortest-path directed acyclic graph (DAG) up to a depth L, which is the maximal length of our sample paths. We then draw a random sample of shortest paths $\mathcal{P}$, capped at a fixed number of paths per source ϕ. A low cap ϕ spreads the sample further over the graph and reduces the correlation between paths that share a source, at the price of a higher computational cost. We draw new source nodes and sample new paths until we reach the target number of paths $N_{\mathcal{P}}$ or no more source nodes can be used under class stratification. As our estimator is free of any model assumption, we choose to sample paths from the shortest-path DAG, rather than the single shortest path tree or the Weisfeiler-Lehman (WL) tree. With a shortest-path DAG there might be different paths that we sample independently for a certain end point of a path v_ℓ. In contrast to the single shortest path tree, the shortest-path DAG preserves the node distribution of the shells with respect to the structure of the propagation graph. Using single shortest paths would prune the number of propagation paths through bottlenecks. The WL tree, on the other hand, is constructed using random walks and therefore mixes nodes of different shells. Even though this is equivalent to most message passing methods in GNNs, it is unsuitable for our approach, as we only want to compare additional information per k-shell.

Finally, we redraw for every shell k a stratified subsample $\mathcal{P}_k^{\text{strat}} \subset \mathcal{P}$ of paths of length at least $\ell \geq k$ to rebalance the classes, so that all per class subsets are equally large per shell $n_{k,c} = n_{k,c'}$ with $n_{k,c} = |\mathcal{P}_{k,c}^{\text{strat}}|$. Due to structural differences, $n_{k,c}$ might not shrink equally with path length for all classes $c \in \mathcal{C}$. This would lead to a new class imbalance $n_{k,c} \neq n_{k,c'}$ in $k > 0$ even if $n_{0,c} = n_{0,c'}$. The subsample keeps as many paths as possible while maintaining class balance, that is, $n_k = \min_c n_{k,c}$ paths per class. We consider only the subset of paths from the previous shell for the subsample of the next shell $\mathcal{P}_k^{\text{strat}} \subset \mathcal{P}_{k-1}^{\text{strat}}$ to preserve the dependency across shells.

5.2 Separability Gap

The largest shell k where the information contained in $\mathcal{P}_k^{\text{strat}}$ is still informative for the task is our estimate of the effective diameter. We quantify the separation of the class-conditional distributions of the nodes within a shell through the pairwise distance of their feature vectors. Let $\boldsymbol{x}_{p_k}$ be the features of the kth node in path $p \in \mathcal{P}_k^{\text{strat}}$. We report the mean squared Euclidean distances over

Algorithm 1 Path Sampling

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, class per node $c(v) \mapsto c_v \in \mathcal{C}$, maximal path length L, maximal number of paths per source node ϕ, maximal total number of paths $N_{\mathcal{P}}$.

$\mathcal{P}, \mathbb{P} \leftarrow \emptyset$
$n_{\mathcal{P}} \leftarrow N_{\mathcal{P}}/|\mathcal{C}|$ # Even path quota per class, summing to $N_{\mathcal{P}}$
for all $c \in \mathcal{C}$ **do**
  $\mathcal{V}^c_{\text{src}} \leftarrow \{v \in \mathcal{V} : c_v = c\}$
  $\mathcal{P}_c \leftarrow \emptyset$
  while $|\mathcal{P}_c| < n_{\mathcal{P}}$ and $\mathcal{V}^c_{\text{src}} \neq \emptyset$ **do**
    $v_{\text{src}} \leftarrow \textsc{Remove}(\mathcal{V}^c_{\text{src}}, 1)$
    $\mathcal{P}_c \leftarrow \mathcal{P}_c \cup \textsc{SampleShortestPaths}(v_{\text{src}}, L, \min(\phi, n_{\mathcal{P}} - |\mathcal{P}_c|))$ # Uniform over all shortest paths
  end while
  $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{P}_c$
end for
$\mathcal{P}_0 \leftarrow \mathcal{P}$
for all $k \in \{0, \dots, L\}$ **do**
  $n_k \leftarrow \min_c |\{p \in \mathcal{P}_k : c(p) = c\}|$
  for all $c \in \mathcal{C}$ **do**
    $\mathcal{P}^{\text{strat}}_{k,c} \leftarrow \textsc{Subsample}(\{p \in \mathcal{P}_k : c(p) = c\}, n_k)$
  end for
  $\mathcal{P}^{\text{strat}}_k \leftarrow \bigcup_{c \in \mathcal{C}} \mathcal{P}^{\text{strat}}_{k,c}$
  $\mathcal{P}_{k+1} \leftarrow \{p \in \mathcal{P}^{\text{strat}}_k : \ell_p \geq k+1\}$
end for
return $\{\mathcal{P}^{\text{strat}}_k : k \in \{0, \dots, L\}\}$

ordered pairs of shell nodes within $W(k)$ and between $B(k)$ classes,

$$W(k) = \frac{\sum_{p \in \mathcal{P}^{\text{strat}}_k} \sum_{\{q \in \mathcal{P}^{\text{strat}}_k : c(p) = c(q), p \neq q\}} \|\boldsymbol{x}_{p_k} - \boldsymbol{x}_{q_k}\|_2^2}{\sum_{c \in \mathcal{C}} n_{k,c}(n_{k,c} - 1)}, \tag{1}$$

$$B(k) = \frac{\sum_{p \in \mathcal{P}^{\text{strat}}_k} \sum_{\{q \in \mathcal{P}^{\text{strat}}_k : c(p) \neq c(q)\}} \|\boldsymbol{x}_{p_k} - \boldsymbol{x}_{q_k}\|_2^2}{N_k^2 - \sum_{c \in \mathcal{C}} n_{k,c}^2}, \tag{2}$$

where the denominators are the numbers of ordered within- and between-class pairs, with the number of paths $n_{k,c} = |\mathcal{P}^{\text{strat}}_{k,c}|$ per class c and in total $N_k = |\mathcal{P}^{\text{strat}}_k|$ that reach the k-shell. We do not have to compute the $N_k \times N_k$ distance matrix. The within-class numerator can be expressed exactly through the deviations of the shell features from their class means, and the between-class numerator additionally through the differences between the class means. This yields both numerators in $\mathcal{O}(N_k d_0)$ time, without any approximation. We give the full derivation in Appendix C. The quantity we use for our diagnosis is the per-shell gap

$$\Delta(k) = \frac{B(k) - W(k)}{W(k)}. \tag{3}$$

The gap quantifies how informative the features of nodes in the k-shell of a node are about the node's target class. If we randomly reassign the labels, both populations of within-class and between-class pairs become exchangeable, and the gap converges to zero. Even though the expected value of Δ is 0 for random feature distributions, the actual measurement is unlikely to exactly reach zero. Therefore, we define the effective diameter

$$\hat{\phi}^{\tau}_{\text{eff}}(\mathcal{G}) = \arg\min_{k \in \mathbb{N}} \Delta(k) \leq \tau,$$

with a threshold τ that determines what level of feature difference is considered as noise.

5.3 Experiments: Reading the effective diameter

We computed the effective diameter on the same real-world datasets we used to investigate oversmoothing in Section 4.1. In Figure 2, we display the results for four of the datasets. Further results

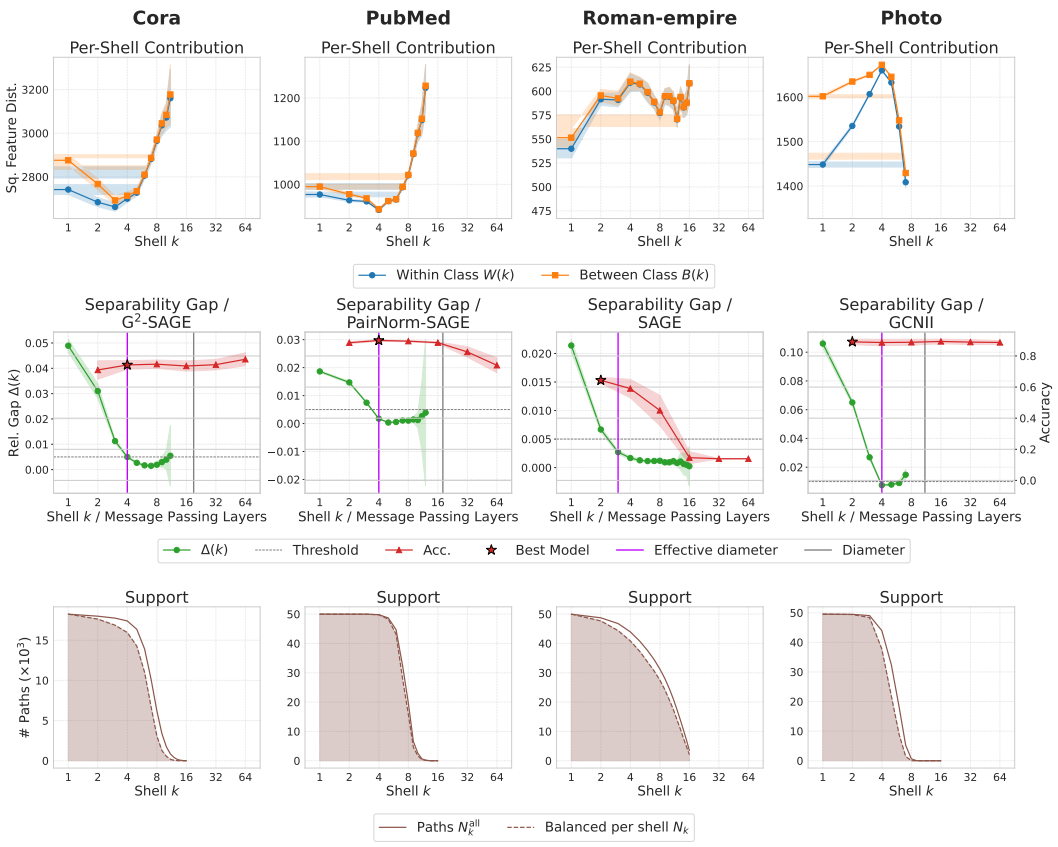

Figure 2: Measurements on the statistics discussed in Sections 5.1 and 5.2. We report results on further datasets in Appendix E. Row 1: Measurements of $W(k)$ and $B(k)$, the bands display the standard deviation obtained using bootstrapping with 20 random samples. Row 2: The relative separability gap $\Delta(k)$ is plotted with the performance of the best model per dataset, the graph diameter (if ¡ 64) and the effective diameter for a threshold $\tau = 0.5\%$ (except for Photo with $\tau = 1\%$). The band of $\Delta(k)$ is the bootstrap standard deviation from Row 1. The band around the performance is the empirical standard deviation from Table 3. The star highlights the shallowest model with a performance within one standard deviation of the overall best model. Row 3: The support for the measurements.

can be found in Appendix E. We sampled paths with a maximum of $N_{\mathcal{P}} = 20\,000$ paths for Cora and $N_{\mathcal{P}} = 50\,000$ paths for the other datasets. Per source node, we sampled a maximum of $\phi = 8$ paths. We set the maximal depth to $L = 16$, which is, for most datasets, sufficient to cover most paths, as we can observe from the support in Row 3 of Figure 2. Roman-Empire is the only dataset that shows any support at this depth. For the others, we set a minimal limit of 100 paths' support. For Cora, we only obtain $N_{k=0} < 19\,000$ after stratification, due to class imbalance.

The within- and between-class node feature distances $W(k)$ and $B(k)$ in the first row of Figure 2 show in all datasets, a gap for small k's, which closes after a few layers. The relative gap $\Delta(k)$, which is displayed in the second row as a green curve, decays correspondingly and stagnates close to 0. This closing gap can be interpreted as the features of nodes in the sampled shells becoming more and more similar. We propose to observe the Δ curve and identify the threshold to cover the stagnation of Δ. We heuristically identified a value of $\tau = 0.005$ as threshold for indicating the stagnation of Δ in all of the investigated datasets except Amazon-Photo, where we set it to $\tau_{\text{PHOTO}} = 0.01$. The effective diameter $\hat{\phi}^{\tau}_{\text{eff}}(\mathcal{G})$ based on the identified threshold is highlighted as a purple line in Figures 2 and 6. We propose to use the Δ curve and $\hat{\phi}^{\tau}_{\text{eff}}(\mathcal{G})$ to set up the search for the optimal number of message passings. The effective diameter can be used as an upper bound,

while the Δ curve should be used to clarify whether later spikes in the separability gap, as shown for the Amazon-Computer dataset in Figure 6, should be considered as possible model depths as well.

In Section 4.1, we observed that even models that mitigate over-smoothing do not perform better with more message passing layers. The effective diameter can offer an explanation for the low optimal model depth. In Row 2 of Figure 2, we plot the depth-accuracy curve of the overall best-performing model per dataset against the curve of the relative gap Δ. We compare the effective diameter $\hat{\phi}^{\tau}_{\text{eff}}(\mathcal{G})$ with the lowest depth where the model performance lies within one standard deviation of the overall best performance. We call this the best model, as additional message passing layers lead to significantly more computational costs without a significant performance improvement. In all cases, top performance can be achieved by a model, i.e., the best model, with a number of message passing layers smaller than or equal to $\hat{\phi}^{\tau}_{\text{eff}}(\mathcal{G})$. The reduced separability gap Δ indicates that additional k-shells offer little or no relevant information to distinguish between the classes of the central nodes. In comparison to the standard graph diameter, which is plotted in grey in Figure 2, the effective diameter provides a more productive guideline for identifying the relevant problem radius.

5.4 Limitations

Even though the effective diameter and the separability gap prove to be effective metrics for judging the relevance of different-sized receptive fields, there are limitations that we discuss in this section.

Identification of τ. The choice of the threshold τ to determine the effective diameter relies on heuristic judgement. We recommend that practitioners observe the Δ curve to identify an appropriate threshold. This seems like a realistic approach to us and more aligned with the diversity of real-world data than to recommend standard elbow detection or gap statistics machinery that may not apply to all observable gap curves.

Task-relevant features. Many structural graph statistics exist that describe characteristics of the graph structure without any model assumptions. However, most of these statistics are purely structural and do not consider attributed graphs. Our statistic is based on both. It samples nodes through propagation paths to represent the structure and then compares the node features. As a consequence, our method is limited to attributed graphs with task-relevant features. On tasks like SAT solving (Selsam et al., 2019; Tönshoff & Grohe, 2026) or the Ising model (Mathys et al., 2026) the feature distance is not meaningful, and one should use existing structural graph descriptors.

Feature interactions. The mean squared Euclidean distance, which is used for the separability gap Δ, does not account for complex feature interactions. This also limits our approach to observing nodes of the shells separately and not combined along the propagation paths. Choosing the Euclidean distance is the result of a trade-off between computational costs and effectiveness. We consider the Euclidean distance as a practical solution to estimate the relevant receptive field in real-world graph datasets. Not modelling feature interactions is in line with our intention to define a statistic that is free of any model assumption.

6 Conclusion

In this paper, we critically examined the practical relevance of over-smoothing in real-world datasets. Our experiments confirm that mitigation techniques prevent over-smoothing and stabilise the performance with respect to model depth. However, deep GNNs still often fail to significantly outperform shallow ones. We conclude that over-smoothing frequently does not limit GNN performance on the benchmark datasets currently established in the Graph Representation Learning literature.

As an alternative explanation for the small optimal depth, we introduced the *effective diameter*, a data-based estimate of the size of the task-relevant receptive field that is free of any model assumptions. It is based on the *shell separability gap*, which compares the feature distances of shell nodes between and within the classes of the central nodes along sampled shortest paths. On all investigated datasets, the separability gap declines within a few hops, and the best-performing models rarely lie beyond the estimated effective diameter. The estimated effective diameter therefore provides a practical, model-agnostic upper bound on the number of message-passing steps required to aggregate task-relevant information. For newly explored datasets, it can thus serve as an indicator of whether over-smoothing is likely to affect GNN performance.

AI USE STATEMENT

In this work, we used generative AI tools to provide critical ingredients for proving mathematical claims, assist in writing proofs, implement methods, and assist with translation. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, and generating synthetic data sets, cleaning and reformating dataset, supporting qualitative and thematic data analysis, interpret results are not applicable to this work. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. LLM-generated aspects in proofs have been reviewed by 2 authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Singh Akansha. Over-squashing in graph neural networks: A comprehensive survey. *Neurocomputing*, 642:130389, 2025.

Adrian Arnaiz-Rodriguez and Federico Errica. Oversmoothing, oversquashing, heterophily, long-range, and more: Demystifying common beliefs in graph machine learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations (ICLR)*, 2022.

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. In *Graph Representation Learning and beyond Workshop, ICML*, 2020.

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

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning (ICML)*, 2020b.

Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, Peter W. Battaglia, Vishal Gupta, Ang Li, Zhongwen Xu, Alvaro Sanchez-Gonzalez, Yujia Li, and Petar Velickovic. ETA prediction with graph neural networks in google maps. In *ACM International Conference on Information & Knowledge Management*, pp. 3767–3776. ACM, October 2021.

Francesco Di Giovanni, James Rowbottom, Benjamin Paul Chamberlain, Thomas Markovich, and Michael M. Bronstein. Understanding convolution on graphs via energies. *Transactions on Machine Learning Research (TMLR)*, 2023, 2023.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations (ICLR)*, 2020.

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 (ICML)*, 2017.

Stephan Günnemann. Graph neural networks: Adversarial robustness. pp. 149–176. 2022. ISBN 978-981-16-6053-5.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Andrew Howard, Ruoming Pang, Hartwig Adam, Quoc V. Le, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, and Yukun Zhu. Searching for MobileNetV3. In *International Conference on Computer Vision (ICCV)*, pp. 1314–1324. IEEE, 2019.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 2015.

Nicolas Keriven. Not too little, not too much: a theoretical analysis of graph (over)smoothing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Nicolas Keriven. Backward Oversmoothing: Why is it hard to train deep Graph Neural Networks? In *International Conference on Machine Learning (ICML)*, 2026.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the Loss Landscape of Neural Nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, November 2018a.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI on Artificial Intelligence*, 2018b.

Shuangli Li, Jingbo Zhou, Tong Xu, Dejing Dou, and Hui Xiong. GeomGCL: Geometric graph contrastive learning for molecular property prediction. *AAAI Conference on Artificial Intelligence*, 2021.

Ping Liu, Haichao Wei, Xiaochen Hou, Jianqiang Shen, Shihai He, Kay Qianqi Shen, Zhujun Chen, Fedor Borisyuk, Daniel Hewlett, Liang Wu, Srikant Veeraraghavan, Alex Tsun, Chengming Jiang, and Wenjing Zhang. LinkSAGE: Optimizing job matching using graph neural networks. In *ACM International Conference on Knowledge Discovery & Data Mining*, 2024.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic GNNs are Strong Baselines: Reassessing GNNs for Node Classification. In *NeurIPS*, 2024.

Joel Mathys, Henrik Christiansen, Federico Errica, Takashi Maruyama, and Francesco Alesiani. LRIM: A Physics-Based Benchmark for Provably Evaluating Long-Range Capabilities in Graph Learning. In *International Conference on Learning Representations (ICLR)*, pp. 43369–43406, 2026.

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*, 2019.

Khang Nguyen, Hieu Nong, Vinh Nguyen, Nhat Ho, Stanley Osher, and Tan Nguyen. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *International Conference on Machine Learning (ICML)*, 2023.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations (ICLR)*, 2020.

MoonJeong Park, Sunghyun Choi, Jaeseung Heo, Eunhyeok Park, and Dongwoo Kim. The oversmoothing fallacy: A misguided narrative in GNN research. In *arXiv*, 2025.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *International Conference on Learning Representations (ICLR)*, 2023.

Andreas Roth and Thomas Liebig. Rank collapse causes over-smoothing and over-correlation in graph neural networks. In *Learning on Graphs Conference (LOG)*, 2024.

T. Konstantin Rusch, Michael M. Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. In *arXiv*, 2023a.

T. Konstantin Rusch, Benjamin P. Chamberlain, Michael W. Mahoney, Michael M. Bronstein, and Siddhartha Mishra. Gradient gating for deep multi-rate learning on graphs. In *International Conference on Learning Representations (ICLR)*, 2023b.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How Does Batch Normalization Help Optimization? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Michael Scholkemper, Xinyi Wu, Ali Jadbabaie, and Michael T. Schaub. Residual connections and normalization can provably prevent oversmoothing in gnns. In *International Conference on Learning Representations (ICLR)*, 2025.

Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a SAT Solver from Single-Bit Supervision. In *International Conference on Learning Representations (ICLR)*. arXiv, March 2019.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, pp. 93–106, 2008.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. In *Relational Representation Learning Workshop, NeurIPS*, volume abs/1811.05868, 2018.

Mingxing Tan and Quoc V Le. EfficientNetV2: Smaller Models and Faster Training. In *International Conference on Machine Learning (ICML)*, pp. 10096–10106, 2021.

Jan Tönshoff and Martin Grohe. Learning from algorithm feedback: One-shot SAT solver guidance with gnns. In *International Conference on Learning Representations (ICLR)*, 2026.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations (ICLR)*, 2022.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Xinyi Wu, Amir Ajorlou, Zihui Wu, and Ali Jadbabaie. Demystifying oversmoothing in attention-based graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning (ICML)*, volume 48, pp. 40–48. JMLR.org, 2016.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *ACM International Conference on Knowledge Discovery & Data Mining*, pp. 974–983. ACM, July 2018.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *International Joint Conference on Artificial Intelligence*, abs/1709.04875, 2017.

Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.

Bingxu Zhang, Changjun Fan, Shixuan Liu, Kuihua Huang, Xiang Zhao, Jincai Huang, and Zhong Liu. The expressive power of graph neural networks: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 37(3):1455–1474, 2025.

Bohang Zhang, Jingchu Gai, Yiheng Du, Qiwei Ye, Di He, and Liwei Wang. Beyond weisfeiler-lehman: A quantitative framework for GNN expressiveness. In *International Conference on Learning Representations (ICLR)*, 2024.

Kaicheng Zhang, Piero Deidda, Desmond Higham, and Francesco Tudisco. Are we measuring oversmoothing in graph neural networks correctly? *International conference on learning representations (ICLR)*, 2026.

Lingxiao Zhao and Leman Akoglu. PairNorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations (ICLR)*, 2020.

Tong Zhao, Yozen Liu, Matthew Kolodner, Kyle Montemayor, Elham Ghazizadeh, Ankit Batra, Zihao Fan, Xiaobin Gao, Xuan Guo, Jiwen Ren, Serim Park, Peicheng Yu, Jun Yu, Shubham Vij, and Neil Shah. GiGL: Large-scale graph neural networks at Snapchat. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.

A Over-smoothing measures

In the literature, several measures for over-smoothing have been proposed. In our experiments, we compute all of these measures to ensure that our conclusions are drawn from a comprehensive analysis.

Most prominently, the discrete Dirichlet energy is used to quantify converging node representations. For a feature matrix with a single feature or hidden dimension $\boldsymbol{x} \in \mathbb{R}^{|\mathcal{V}|\times 1}$, its vectorised form is defined as

$$E(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x} = \frac{1}{2}\sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{V}|} a_{ij}\left(x_i - x_j\right)^2,$$

where $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ denotes the Laplacian matrix with degree matrix $\boldsymbol{D}$. In multidimensional space, the Dirichlet energy is defined for a continuous function $u(\cdot)$ using the Euclidean norm $\|\cdot\|_2$ over all dimensions (channels) $c \in C$ by

$$E(u) = \frac{1}{2}\int_{\Omega} \|\nabla u\|_2^2 dx.$$

In our case, this corresponds to summing the Dirichlet energy over all feature dimensions, with $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 = (x_{ic} - x_{jc})^2 + \cdots + (x_{iC} - x_{jC})^2$. Since we compare the Dirichlet energy of hidden features across models with potentially different optimal feature dimensions, we normalise by the feature dimension to ensure comparability:

$$E(\boldsymbol{X}) = \frac{1}{C}\mathbf{1}^T \mathrm{diag}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X}) = \frac{1}{2C}\sum_{c=1}^{C}\sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{V}|} a_{ij}\left(x_{ic} - x_{jc}\right)^2, \tag{4}$$

Rusch et al. (2023a) define a slight variation of the Dirichlet energy for measuring over-smoothing on graphs, normalising by the number of nodes rather than by two. They also apply a square root to the Dirichlet energy. Note that both the normalisation and the square root merely rescale the values and do not affect the expected exponential decay towards zero. We therefore adopt the definition in Equation (4).

The Dirichlet energy is equal to zero if and only if all node representations are identical. However, this behaviour only arises after over-smoothing, for example, for GAT layers (Wu et al., 2023; Cai & Wang, 2020). For GCNs, over-smoothed representations are expected to be scaled by the node

degree (Oono & Suzuki, 2020). Consequently, Cai & Wang (2020) propose a degree-normalised Dirichlet energy that converges to zero for over-smoothing in GCNs. Extending this to our multidimensional formulation yields

$$E_D(\boldsymbol{X}) = \frac{1}{C}\sum_{c=1}^{C}\sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{V}|} a_{ij}\left(\frac{x_{ic}}{\sqrt{1+d(i)}} - \frac{x_{jc}}{\sqrt{1+d(j)}}\right)^2. \tag{5}$$

Both versions of the Dirichlet energy can be made invariant to the feature scale by normalising as proposed by Di Giovanni et al. (2023)

$$E^{\text{norm}}(\boldsymbol{X}) = \frac{E(\boldsymbol{X})}{\|\boldsymbol{X}\|_F^2}, \quad E_{\text{D}}^{\text{norm}}(\boldsymbol{X}) = \frac{E_D(\boldsymbol{X})}{\|\boldsymbol{X}\|_F^2},$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

Wu et al. (2023) define a node similarity measure based on deviations from the global mean feature vector:

$$\mu_{\text{WU}}(\boldsymbol{X}) = \left\|\boldsymbol{X} - \frac{\mathbf{1}\mathbf{1}^T\boldsymbol{X}}{|\mathcal{V}|}\right\|_F. \tag{6}$$

This metric also converges to zero in cases where GAT-based models exhibit over-smoothing.

Chen et al. (2020a) introduce the Mean Average Distance (MAD), a smoothness metric based on the cosine similarity between pairs of nodes within a target mask. In our work, we compute only the neighbourhood smoothness, defined for a given k-hop neighbourhood as

$$\mu_{\text{MAD}}^{(k)}(\boldsymbol{X}) = \frac{1}{|\mathcal{E}^{(k)}|}\sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{V}|} a_{ij}^{(k)}\left(1 - \frac{\boldsymbol{x}_i\boldsymbol{x}_j}{\|\boldsymbol{x}_i\|_2\|\boldsymbol{x}_j\|_2}\right),$$

where $\mathcal{E}^{(k)}$ and $a_{ij}^{(k)}$ are the edge set and entry of the adjacency matrix of the k-hop neighbourhood $\mathcal{N}^{(:k)}(\cdot)$, respectively.

Beyond similarity-based measures, Zhang et al. (2026) propose quantifying over-smoothing via a continuous relaxation of the (numerical) rank of the feature representation matrix, capturing convergence toward an arbitrary low-dimensional subspace rather than requiring exact alignment with a single dominant eigenvector. Given singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min\{|\mathcal{V}|,C\}}$ of the feature matrix $\boldsymbol{X}$, the numerical rank is defined as

$$\text{NumRank}(\boldsymbol{X}) = \frac{\|\boldsymbol{X}\|_F^2}{\|\boldsymbol{X}\|_2^2}. \tag{7}$$

The effective rank instead interprets the normalised singular values $p_k = \sigma_k / \sum_i \sigma_i$ as a probability distribution over the singular spectrum and takes its exponentiated Shannon entropy:

$$\text{Erank}(\boldsymbol{X}) = \exp\left(-\sum_k p_k \log p_k\right). \tag{8}$$

Both NumRank and Erank satisfy $1 \leq \text{NumRank}(\boldsymbol{X}), \text{Erank}(\boldsymbol{X}) \leq \min\{|\mathcal{V}|, C\}$, converging to one as the feature representations collapse toward a rank-one matrix.

In this work, we evaluate all discussed over-smoothing measures on the hidden node embeddings after the final message passing and update step. We compute each measure for every random test initialisation, and for all nodes of the graph, not only those in the test set. In Figure 1 and in Figures 3 to 5, we report the measures theoretically appropriate for the respective model type: the degree-normalised Dirichlet energy from Equation (5) and its feature normalised version for models

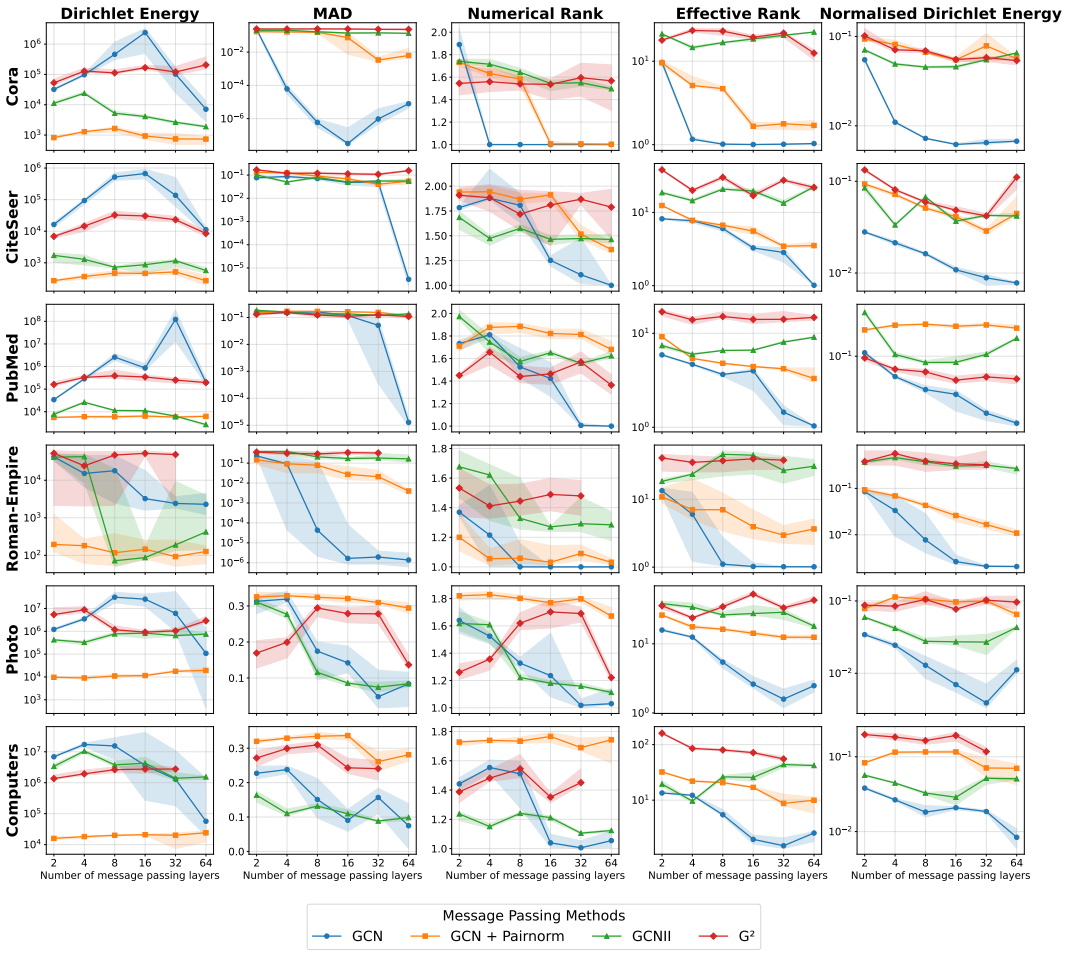

Figure 3: Dirichlet energy and MAD with a 3-hop neighbourhood of the whole dataset for all tested models with GCN message passing and datasets at different depths after hyperparameter tuning and training. The error bands represent the 25- and 75-percentiles.

based on GCN message passing, and the Dirichlet energy from Equation (4) and the node similarity measure from Equation (6) for models using GATv2 or GraphSAGE message passing. Further, we evaluate the MAD with $k = 3$, and the rank-based measure on all models.

A remark on the results of the over-smoothing measures. In Figure 1, we observe that the Dirichlet energy for the GCN does not exhibit the expected exponential decrease reported in other works (Rusch et al., 2023a; Park et al., 2025). These different experimental results may arise from conducting hyperparameter tuning and model training for the experimental study in this work. These findings are consistent with empirical works that show that a simple scaling of the weight matrices (Arnaiz-Rodriguez & Errica, 2026; Roth & Liebig, 2024) prevents a collapse of the Dirichlet energy.

B Experimental Details

The empirical results discussed in Section 4.1 were obtained through an extensive set of experiments, which will be further detailed in this appendix.

We evaluated methods against over-smoothing on six datasets from different domains and of varying sizes. We performed experiments on the citation datasets Cora, Citeseer, and PubMed Sen et al. (2008), as these are the benchmarks on which most over-smoothing mitigation meth-

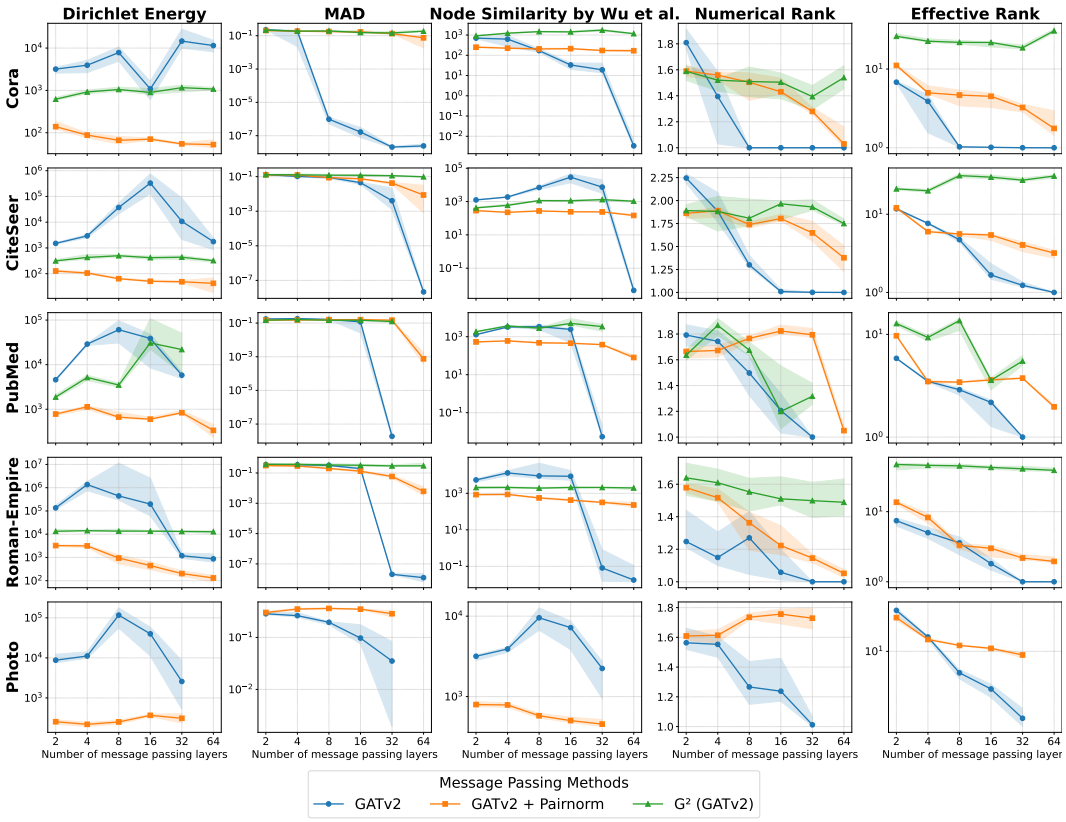

Figure 4: Dirichlet energy, MAD with a 3-hop neighbourhood and node similarity by Wu et al. (2023) of the whole dataset for all tested models with GATv2 message passing and datasets at different depths after hyperparameter tuning and training. The error bands represent the 25- and 75-percentiles.

ods have been developed. To further strengthen our experimental conclusions, we additionally included the citation network OGBN-arXiv (Hu et al., 2020), which is orders of magnitude larger, the heterophilic Roman-Empire text graph (Platonov et al., 2023), and the Photo and Computers co-purchase graphs from the Amazon dataset (Shchur et al., 2018). We did not include any datasets with graph-level tasks for the over-smoothing experiments, as this phenomenon has primarily been studied in the context of node-level tasks. For graph-level tasks, a performance degradation due to the convergence of node representations is also not expected, since hidden node embeddings are aggregated into a single graph representation after message passing.

For each model, we optimised the hyperparameters on each dataset and for each number of message passing layers $\ell \in \{2, 4, 8, 16, 32, 64\}$. The common hyperparameters, together with their search spaces or fixed values used for all models, are summarised in Table 2. For GCNII and G^2, the search spaces are not dataset-specific. We searched over $\alpha \in \{0.1, 0.3, 0.5\}$ and $\lambda \in \{0.5, 1.0, 1.5\}$ for GCNII, following Chen et al. (2020b), and over the exponent $p \in \{1, 3, 5\}$ for G^2, following Rusch et al. (2023b).

The hyperparameter optimisation and testing procedures were implemented in accordance with the corresponding literature:

- **Cora, Citeseer, and PubMed.** For the three citation networks, we used the publicly available "planetoid" split (Yang et al., 2016). For hyperparameter optimisation, we combined the train and validation sets and performed a 10-fold cross-validation. The test set remained unchanged.

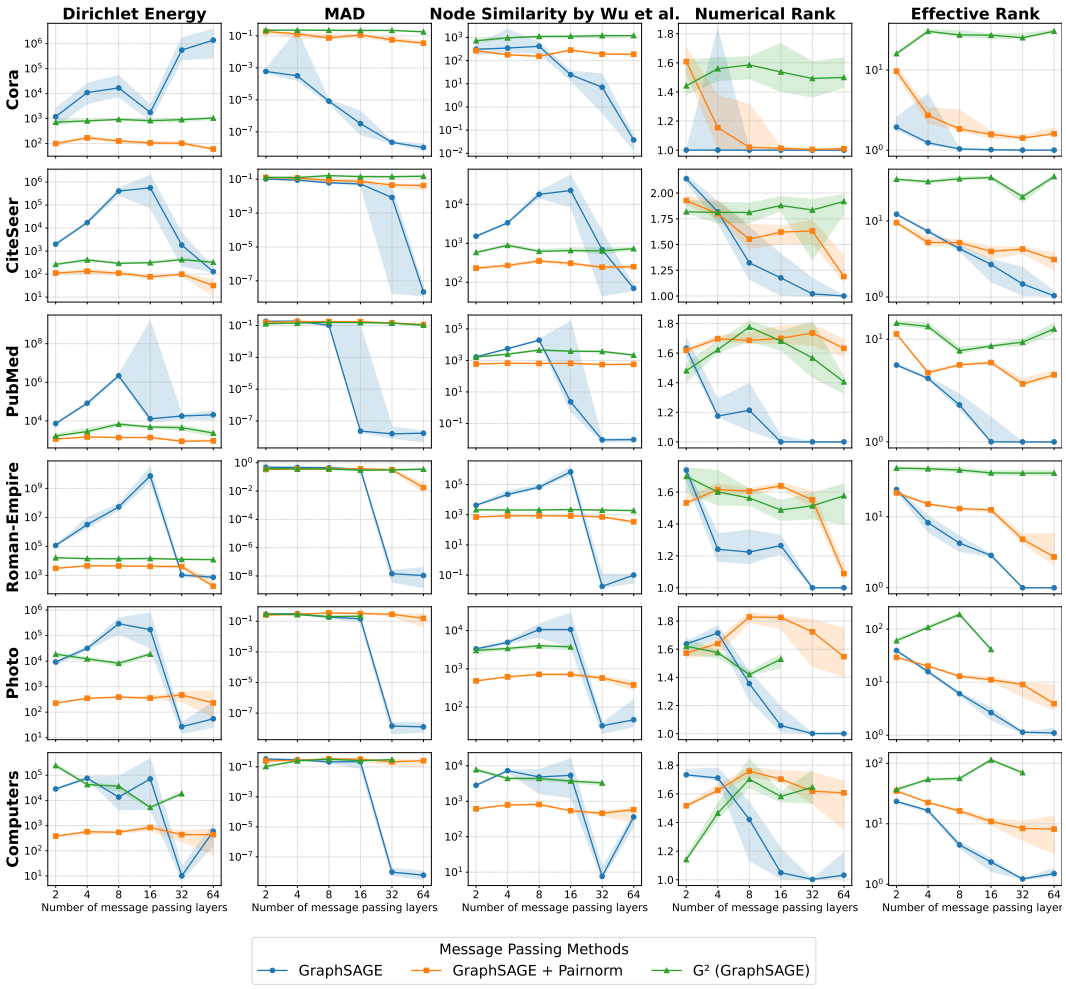

Figure 5: Dirichlet energy, MAD with a 3-hop neighbourhood and node similarity by Wu et al. (2023) of the whole dataset for all tested models with GraphSAGE message passing and datasets at different depths after hyperparameter tuning and training. The error bands represent the 25- and 75-percentiles.

- **Roman-Empire.** For the heterophilic Roman-Empire dataset, we used the publicly available split consisting of 10 random 50%/25%/25% train/validation/test splits (Platonov et al., 2023). We adjusted the testing procedure slightly and trained and tested the model with the optimal hyperparameter setting per fold with three random parameter initialisations as suggested by Errica et al. (2020).

- **Amazon.** For the Amazon Photo and Computers datasets, we applied the 20/30/rest split described in Shchur et al. (2018). Since the original splits are not publicly available, we implemented the sampling strategy, selecting 20 and 30 samples per class for the train and validation sets, respectively. For hyperparameter optimisation, we resampled eight stratified folds from the union of train and validation sets and conducted cross-validation. The test set from the original 20/30/rest split was kept fixed.

For the datasets for which we did not apply the Errica protocol, we evaluated each model on the test set using the hyperparameter configuration that achieved the best mean validation loss across all folds, and we report the mean over 10 random parameter initialisations.

Table 2: Search space or fixed values for standard hyperparameters defined for each dataset applied to every model; RoP(p, lr, f, lr_{min}) denotes a starting learning rate lr to be reduced by factor f when the validation loss plateaus with patience p down to a minimum of lr_{min}.

DATASET	HIDDEN SIZE	DROPOUT	LEARNING RATE	EPOCHS (MAX)	PATIENCE	BATCH SIZE
CORA	{32, 64, 128}	{0.2, 0.4, 0.6, 0.8}	0.001	500	30	1
CITESEER	{32, 64, 128}	{0.2, 0.4, 0.6, 0.8}	0.001	500	30	1
PUBMED	{32, 64, 128}	{0.2, 0.4, 0.6, 0.8}	0.001	500	30	1
ROMAN-EMPIRE	{32, 64, 128}	{0.2, 0.4, 0.6, 0.8}	0.001	500	30	1
PHOTO	{256, 512, 1024}	{0.0, 0.1, 0.2, 0.4}	RoP(10, 0.001, 0.5, 2,4E-4)	3000	20	1
COMPUTERS	{256, 512, 1024}	{0.0, 0.1, 0.2, 0.4}	RoP(10, 0.001, 0.5, 2,4E-4)	3000	20	1

C COMPUTING THE WITHIN- AND BETWEEN-CLASS DISTANCES

We define the numerators of $W(k)$ in Equation (1) and $B(k)$ in Equation (2) as pairwise feature distances between path elements of the same class and of different classes, which results in a computational complexity of $\mathcal{O}(N^2 d_0)$, as we have to evaluate the full $N_k \times N_k$ matrix of pairwise distances. However, we can reformulate the numerator of $W(k)$ as the feature distance of path elements of one class to their mean and the numerator of $B(k)$ as the distance between class means, and thus obtain a computational complexity of $\mathcal{O}(N d_0)$ as we show in this Appendix.

Throughout this appendix, the shell k is fixed and we drop it from the notation where it is unambiguous. The class means, the shell mean and the traces of the biased covariances are

$$\mu_c = \frac{1}{n_c} \sum_{p \in \mathcal{P}_c^{\text{strat}}} \boldsymbol{x}_p, \qquad \bar{\boldsymbol{x}} = \frac{1}{N} \sum_{p \in \mathcal{P}^{\text{strat}}} \boldsymbol{x}_p,$$

$$\operatorname{tr} \hat{\Sigma}_c = \frac{1}{n_c} \sum_{p \in \mathcal{P}_c^{\text{strat}}} \|\boldsymbol{x}_p - \mu_c\|_2^2, \quad \operatorname{tr} \hat{\Sigma} = \frac{1}{N} \sum_{p \in \mathcal{P}^{\text{strat}}} \|\boldsymbol{x}_p - \bar{\boldsymbol{x}}\|_2^2.$$

Lemma 1. *Let $\mathcal{X}$ be a finite multiset of M vectors in $\mathbb{R}^{d_0}$ with mean $\boldsymbol{m}$, and let $\boldsymbol{a} \in \mathbb{R}^{d_0}$. Then*

$$\sum_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{a}\|_2^2 = \sum_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{m}\|_2^2 + M \|\boldsymbol{m} - \boldsymbol{a}\|_2^2.$$

Proof. Writing $\boldsymbol{x} - \boldsymbol{a} = (\boldsymbol{x} - \boldsymbol{m}) + (\boldsymbol{m} - \boldsymbol{a})$ and expanding the square gives

$$\sum_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{a}\|_2^2 = \sum_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{m}\|_2^2 + 2(\boldsymbol{m} - \boldsymbol{a})^T \sum_{\boldsymbol{x} \in \mathcal{X}} (\boldsymbol{x} - \boldsymbol{m}) + M \|\boldsymbol{m} - \boldsymbol{a}\|_2^2.$$

The cross term vanishes, since $\sum_{\boldsymbol{x} \in \mathcal{X}} (\boldsymbol{x} - \boldsymbol{m}) = M\boldsymbol{m} - M\boldsymbol{m} = \boldsymbol{0}$. □

Lemma 2. *Let $\mathcal{X}$ and $\mathcal{Y}$ be finite multisets of $M_\mathcal{X}$ and $M_\mathcal{Y}$ vectors with means $\boldsymbol{m}_\mathcal{X}$ and $\boldsymbol{m}_\mathcal{Y}$. Then*

$$\sum_{\boldsymbol{x} \in \mathcal{X}} \sum_{\boldsymbol{y} \in \mathcal{Y}} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 = M_\mathcal{Y} \sum_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{m}_\mathcal{X}\|_2^2 + M_\mathcal{X} \sum_{\boldsymbol{y} \in \mathcal{Y}} \|\boldsymbol{y} - \boldsymbol{m}_\mathcal{Y}\|_2^2 + M_\mathcal{X} M_\mathcal{Y} \|\boldsymbol{m}_\mathcal{X} - \boldsymbol{m}_\mathcal{Y}\|_2^2.$$

Proof. For a fixed $\boldsymbol{y}$, Lemma 1 applied to $\mathcal{X}$ with $\boldsymbol{a} = \boldsymbol{y}$ gives $\sum_{\boldsymbol{x}} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 = \sum_{\boldsymbol{x}} \|\boldsymbol{x} - \boldsymbol{m}_\mathcal{X}\|_2^2 + M_\mathcal{X} \|\boldsymbol{m}_\mathcal{X} - \boldsymbol{y}\|_2^2$. Summing over $\boldsymbol{y} \in \mathcal{Y}$ yields

$$\sum_{\boldsymbol{x} \in \mathcal{X}} \sum_{\boldsymbol{y} \in \mathcal{Y}} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 = M_\mathcal{Y} \sum_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{m}_\mathcal{X}\|_2^2 + M_\mathcal{X} \sum_{\boldsymbol{y} \in \mathcal{Y}} \|\boldsymbol{y} - \boldsymbol{m}_\mathcal{X}\|_2^2.$$

Applying Lemma 1 once more to $\mathcal{Y}$ with $\boldsymbol{a} = \boldsymbol{m}_\mathcal{X}$ gives $\sum_{\boldsymbol{y}} \|\boldsymbol{y} - \boldsymbol{m}_\mathcal{X}\|_2^2 = \sum_{\boldsymbol{y}} \|\boldsymbol{y} - \boldsymbol{m}_\mathcal{Y}\|_2^2 + M_\mathcal{Y} \|\boldsymbol{m}_\mathcal{Y} - \boldsymbol{m}_\mathcal{X}\|_2^2$, which completes the proof. □

Proposition 1 (Within-class distance)**.** *The numerator of Equation* (1) *depends only on the deviations of the shell features from their class means,*

$$\sum_{p \in \mathcal{P}_k^{\text{strat}}} \sum_{\{q \in \mathcal{P}_k^{\text{strat}} : c(p) = c(q), p \neq q\}} \|\boldsymbol{x}_p - \boldsymbol{x}_q\|_2^2 = 2 \sum_{c \in \mathcal{C}} n_c \sum_{p \in \mathcal{P}_c^{\text{strat}}} \|\boldsymbol{x}_p - \mu_c\|_2^2 = 2 \sum_{c \in \mathcal{C}} n_c^2 \operatorname{tr} \hat{\Sigma}_c.$$

Proof. The pairs with $p = q$ have distance zero, so the condition $p \neq q$ can be dropped without changing the sum. The remaining pairs with $c(p) = c(q)$ decompose into the pairs within each class, so the numerator equals $\sum_{c\in\mathcal{C}} \sum_{p\in\mathcal{P}_c^{\text{strat}}} \sum_{q\in\mathcal{P}_c^{\text{strat}}} \|\boldsymbol{x}_p - \boldsymbol{x}_q\|_2^2$. Lemma 2 with $\mathcal{X} = \mathcal{Y} = \{\boldsymbol{x}_p : p \in \mathcal{P}_c^{\text{strat}}\}$ has $\boldsymbol{m}_\mathcal{X} = \boldsymbol{m}_\mathcal{Y} = \mu_c$, so the term of the mean difference vanishes and each class contributes $2n_c \sum_{p\in\mathcal{P}_c^{\text{strat}}} \|\boldsymbol{x}_p - \mu_c\|_2^2 = 2n_c^2 \operatorname{tr} \hat{\Sigma}_c$. □

Proposition 2 (Between-class distance)**.** *The numerator of Equation* (2) *depends on the deviations of the shell features from their class means and on the differences between the class means,*

$$\sum_{p\in\mathcal{P}_k^{\text{strat}}} \sum_{\{q\in\mathcal{P}_k^{\text{strat}}: c(p)\neq c(q)\}} \|\boldsymbol{x}_p - \boldsymbol{x}_q\|_2^2 = 2\sum_{c\in\mathcal{C}} n_c(N-n_c) \operatorname{tr} \hat{\Sigma}_c + \sum_{c\in\mathcal{C}} \sum_{c'\in\mathcal{C}\setminus\{c\}} n_c n_{c'} \|\mu_c - \mu_{c'}\|_2^2 .$$

Proof. The ordered between-class pairs decompose into the ordered pairs of distinct classes (c, c'). Lemma 2 with $\mathcal{X} = \{\boldsymbol{x}_p : p \in \mathcal{P}_c^{\text{strat}}\}$ and $\mathcal{Y} = \{\boldsymbol{x}_q : q \in \mathcal{P}_{c'}^{\text{strat}}\}$ gives for each of them

$$\sum_{p\in\mathcal{P}_c^{\text{strat}}} \sum_{q\in\mathcal{P}_{c'}^{\text{strat}}} \|\boldsymbol{x}_p - \boldsymbol{x}_q\|_2^2 = n_c n_{c'} \operatorname{tr} \hat{\Sigma}_c + n_c n_{c'} \operatorname{tr} \hat{\Sigma}_{c'} + n_c n_{c'} \|\mu_c - \mu_{c'}\|_2^2 .$$

Summing over all ordered pairs $c \neq c'$, the first two terms both give $\sum_c n_c \operatorname{tr} \hat{\Sigma}_c \sum_{c'\neq c} n_{c'} = \sum_c n_c(N - n_c) \operatorname{tr} \hat{\Sigma}_c$, and the third term gives the sum over the class mean differences. □

D Report of extended over-smoothing experiments

In Table 3 we report the results for all experiments, including additionally GATv2 and GraphSAGE as message passing methods.

E Report of extended effective diameter experiments

In Figure 6, we report the results for experiments on the effective diameter for all datasets.

Table 3: Full table of experiments on node-level tasks with all tested models, comprising GCN and GATv2 and GraphSAGE in combination with tested mitigation techniques for over-smoothing for a range of model depths. The accuracy is given in % with the standard deviation in brackets. The best-performing model is highlighted in bold.

Dataset	Model	Layers					
		2	4	8	16	32	64
Cora	GCN	**74.1 (2.4)**	36.1 (13.4)	31.9 (0.0)	31.9 (0.0)	31.9 (0.0)	31.9 (0.0)
	GCN + Pairnorm	**66.2 (6.4)**	61.7 (6.3)	59.6 (3.7)	38.5 (7.9)	33.0 (3.4)	34.2 (4.9)
	GCNII	85.0 (0.3)	84.9 (2.0)	84.4 (2.4)	84.6 (3.0)	84.6 (3.5)	**86.8 (0.6)**
	G^2	70.5 (6.1)	74.7 (0.5)	73.1 (2.5)	74.7 (0.7)	**74.9 (0.6)**	70.7 (5.9)
	GATv2	**72.4 (5.5)**	53.9 (17.8)	31.9 (0.0)	31.9 (0.0)	31.9 (0.0)	31.9 (0.0)
	GATv2 + Pairnorm	**68.9 (4.9)**	61.5 (6.2)	57.5 (14.9)	61.6 (7.0)	52.4 (5.7)	41.4 (7.8)
	G^2 (GATv2)	74.6 (3.0)	75.5 (1.1)	75.5 (1.0)	75.4 (2.7)	**76.4 (3.7)**	75.9 (7.0)
	GraphSAGE	31.9 (0.0)	**47.7 (20.5)**	31.9 (0.0)	31.9 (0.0)	31.9 (0.0)	31.9 (0.0)
	GraphSAGE + Pairnorm	**65.2 (7.9)**	43.6 (10.1)	39.0 (8.9)	39.9 (5.5)	36.1 (5.4)	33.9 (4.4)
	G^2 (GraphSAGE)	71.0 (5.9)	74.2 (2.8)	74.7 (2.7)	73.5 (3.3)	74.4 (3.3)	**77.9 (4.2)**
CiteSeer	GCN	**75.5 (0.8)**	74.4 (0.4)	73.6 (0.7)	65.6 (8.0)	60.7 (5.3)	18.1 (0.0)
	GCN + Pairnorm	**68.3 (0.9)**	67.4 (1.5)	66.6 (4.2)	65.0 (5.5)	44.5 (7.2)	29.2 (6.6)
	GCNII	77.5 (0.6)	77.5 (0.6)	**78.1 (0.6)**	77.7 (0.5)	77.4 (1.1)	77.9 (0.6)
	G^2	75.0 (3.7)	74.4 (5.1)	**76.3 (0.6)**	73.6 (5.3)	75.8 (0.4)	73.9 (5.3)
	GATv2	**74.9 (0.7)**	74.3 (0.6)	72.7 (0.8)	68.8 (5.5)	38.1 (15.7)	21.6 (2.4)
	GATv2 + Pairnorm	**70.2 (1.1)**	67.6 (0.7)	68.2 (3.1)	68.0 (5.5)	52.3 (7.1)	26.1 (5.8)
	G^2 (GATv2)	74.3 (5.0)	74.2 (5.1)	**76.2 (0.7)**	75.8 (0.8)	75.9 (0.8)	75.4 (0.7)
	GraphSAGE	**75.2 (0.5)**	74.1 (0.4)	72.5 (0.8)	58.7 (10.8)	32.8 (15.1)	19.6 (1.7)
	GraphSAGE + Pairnorm	**65.4 (4.9)**	58.8 (7.5)	60.8 (8.5)	55.8 (8.1)	50.1 (6.1)	27.2 (6.4)
	G^2 (GraphSAGE)	75.5 (3.9)	76.4 (0.5)	75.4 (3.7)	**76.6 (0.5)**	74.0 (5.0)	76.6 (0.5)
PubMed	GCN	**87.3 (0.4)**	85.8 (0.2)	83.9 (3.5)	80.4 (6.0)	55.9 (14.6)	40.7 (0.2)
	GCN + Pairnorm	**88.2 (0.4)**	86.5 (0.3)	85.7 (0.4)	84.5 (0.3)	84.5 (0.5)	76.5 (5.2)
	GCNII	88.0 (0.5)	87.9 (0.4)	87.8 (0.4)	87.8 (0.5)	88.0 (0.4)	**88.6 (0.4)**
	G^2	**88.2 (0.3)**	86.9 (1.4)	86.2 (2.7)	87.4 (0.6)	84.4 (6.6)	82.2 (14.4)
	GATv2	**87.8 (0.4)**	87.1 (0.4)	82.6 (4.5)	65.9 (17.7)	44.0 (10.3)	OOM
	GATv2 + Pairnorm	**86.7 (0.5)**	85.9 (0.4)	84.7 (0.4)	84.5 (0.3)	83.1 (0.7)	40.7 (0.1)
	G^2 (GATv2)	**87.8 (0.6)**	86.9 (0.5)	86.9 (1.2)	76.5 (10.2)	73.4 (12.7)	OOM
	GraphSAGE	87.4 (0.5)	**88.0 (0.4)**	77.0 (6.1)	53.8 (16.7)	40.7 (0.0)	40.8 (0.2)
	GraphSAGE + Pairnorm	88.5 (0.6)	**90.0 (0.5)**	89.5 (0.4)	88.6 (0.5)	82.6 (3.3)	74.0 (4.8)
	G^2 (GraphSAGE)	**88.0 (0.2)**	87.5 (0.4)	86.4 (0.7)	86.8 (0.3)	85.2 (3.4)	82.2 (6.7)
Roman-Empire	GCN	**35.7 (8.7)**	22.1 (8.5)	15.0 (2.0)	14.5 (1.2)	13.9 (0.1)	14.0 (0.3)
	GCN + Pairnorm	**17.8 (2.7)**	16.6 (1.5)	16.3 (1.5)	15.5 (2.6)	14.7 (1.1)	13.9 (0.5)
	GCNII	55.1 (10.2)	**57.7 (13.7)**	43.4 (13.7)	35.5 (12.9)	38.6 (15.2)	38.7 (13.8)
	G^2	52.2 (14.5)	54.3 (12.8)	52.8 (13.5)	**54.9 (12.7)**	54.4 (11.7)	OOM
	GATv2	**34.5 (9.3)**	34.3 (10.7)	23.7 (11.1)	14.6 (2.3)	13.9 (0.1)	13.9 (0.1)
	GATv2 + Pairnorm	**23.8 (13.3)**	16.9 (3.6)	15.9 (2.3)	15.1 (2.0)	14.5 (0.9)	13.7 (0.7)
	G^2 (GATv2)	53.2 (12.4)	51.4 (11.6)	**54.4 (10.3)**	52.1 (10.4)	50.6 (8.9)	OOM
	GraphSAGE	**64.4 (1.9)**	59.0 (5.9)	45.0 (9.7)	14.6 (3.9)	13.9 (0.1)	13.9 (0.1)
	GraphSAGE + Pairnorm	53.9 (11.0)	**55.9 (14.1)**	41.3 (8.8)	42.7 (11.6)	23.4 (10.5)	13.8 (1.0)
	G^2 (GraphSAGE)	54.4 (13.6)	**56.6 (10.2)**	55.7 (10.3)	54.0 (10.2)	54.6 (10.3)	OOM
Photo	GCN	**87.1 (0.9)**	85.1 (1.4)	71.7 (14.8)	49.6 (20.3)	27.5 (18.7)	23.0 (16.8)
	GCN + Pairnorm	**88.0 (1.8)**	82.9 (2.4)	84.0 (2.9)	84.5 (2.6)	83.5 (1.9)	74.2 (9.1)
	GCNII	89.1 (1.8)	88.6 (2.1)	88.8 (1.9)	**89.3 (1.5)**	88.9 (1.6)	88.7 (1.2)
	G^2	71.6 (15.0)	83.2 (6.8)	86.4 (3.4)	82.6 (8.2)	**86.9 (2.6)**	74.8 (10.5)
	GATv2	**82.7 (2.3)**	82.7 (4.2)	71.9 (9.7)	58.2 (21.9)	21.7 (9.7)	OOM
	GATv2 + Pairnorm	80.5 (2.4)	82.2 (3.3)	81.1 (4.0)	**84.9 (2.5)**	74.3 (18.2)	OOM
	GraphSAGE	**88.4 (1.9)**	86.3 (1.2)	63.2 (13.1)	47.3 (24.4)	14.2 (9.6)	11.7 (8.0)
	GraphSAGE + Pairnorm	**86.8 (1.9)**	78.4 (5.2)	85.4 (2.5)	81.8 (2.5)	61.3 (22.0)	41.7 (28.6)
	G^2 (GraphSAGE)	**84.3 (2.6)**	82.7 (3.9)	79.5 (7.9)	71.8 (18.9)	OOM	OOM
Computers	GCN	**76.9 (4.0)**	71.4 (10.6)	36.9 (21.2)	28.6 (16.3)	32.2 (13.5)	15.6 (16.3)
	GCN + Pairnorm	**81.8 (2.5)**	77.9 (3.1)	78.5 (3.0)	75.3 (4.1)	47.2 (24.6)	50.0 (16.7)
	GCNII	**82.4 (1.6)**	80.5 (2.1)	80.3 (2.2)	80.3 (2.0)	81.0 (1.8)	81.2 (1.9)
	G^2	70.7 (2.9)	71.6 (3.2)	**73.7 (2.0)**	70.9 (3.7)	72.8 (7.6)	OOM
	GraphSAGE	**79.2 (4.3)**	75.8 (5.0)	23.1 (20.5)	19.2 (12.5)	3.1 (1.6)	10.7 (5.6)
	GraphSAGE + Pairnorm	**82.1 (1.5)**	79.7 (6.4)	76.6 (9.3)	58.0 (25.5)	36.2 (23.3)	37.5 (20.4)
	G^2 (GraphSAGE)	70.8 (5.9)	72.8 (7.2)	73.8 (8.6)	**74.5 (8.2)**	74.3 (4.5)	OOM

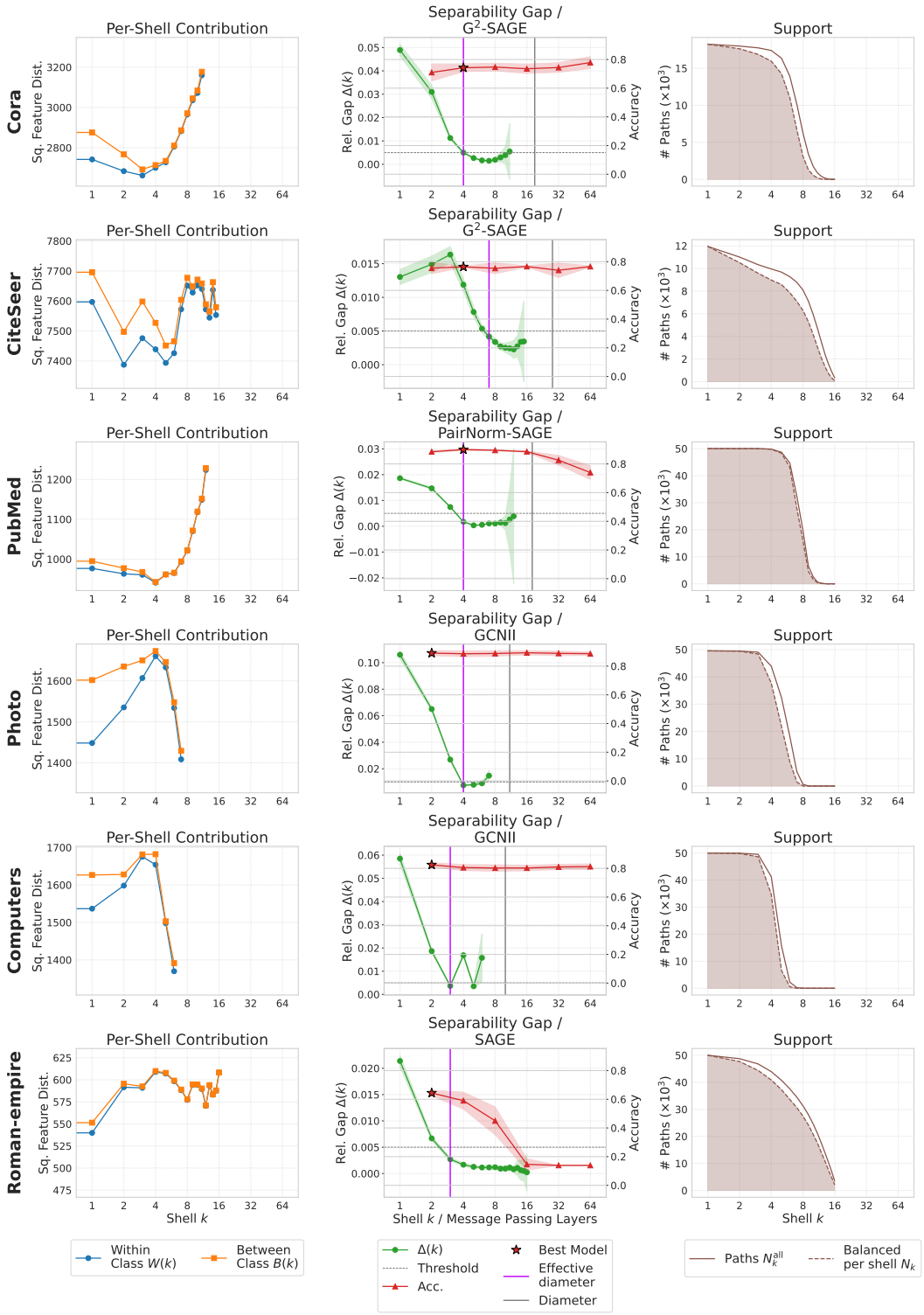

Figure 6: Measurements on the statistics discussed in Sections 5.1 and 5.2. We report results on further datasets in Appendix E. Row 1: Measurements of $W(k)$ and $B(k)$, the bands display the standard deviation obtained using bootstrapping with 20 random samples. Row 2: The relative separability gap $\Delta(k)$ is plotted with the performance of the best model per dataset, the graph diameter (if ¡ 64) and the effective diameter for a threshold $\tau = 0.5\%$ (except for Photo with $\tau = 1\%$). The band of $\Delta(k)$ is the bootstrap standard deviation from Row 1. The band around the performance is the empirical standard deviation from Table 3. The star highlights the shallowest model with a performance within one standard deviation of the overall best model. Row 3: The support for the measurements.