Mitigating Oversmoothing in GNNs on Heterophilic Graphs: Evaluating GCNII and DropEdge

Atharva Singh*
atharva@atharvasingh.com

Mahi H Shah*
mahi.shah@podarinternationalschool.com

*These authors contributed equally to this work.
This project was developed as part of the BeyondAI programme by ThinkingBeyond Education, with support and mentorship that substantially informed its direction, including contributions from Dr. Filip Bar and Dr. Helena Bahrami.

Abstract

Graph neural networks are increasingly deployed in real-world settings where homophily cannot be assumed, making robustness to oversmoothing a critical concern. This paper evaluates the effectiveness of two prominent oversmoothing mitigation techniques, GCNII and DropEdge, under heterophilic graph conditions. While such methods are originally validated on homophilic citation networks, their behavior in heterophilic regimes remains underexplored. An empirical comparison was conducted across increasing depths on the heterophilic WebKB datasets, using GCN and GraphSAGE backbones, with classification accuracy used as the main evaluation metric. Results indicate that oversmoothing emerges earlier and more severely for DropEdge in heterophilic graphs, leading to pronounced performance degradation as depth increases. In contrast, GCNII maintained relatively stable performance at greater depths.

1 Introduction

Graph Neural Networks (GNNs) are a class of deep learning architectures designed to operate directly on graph-structured data. Formally, a GNN can be viewed as an optimizable transformation over all components of a graph (its node features, edge attributes, and global structural context) while respecting core graph symmetries [1].

Graphs provide an exceptionally versatile modelling framework, capable of representing a wide spectrum of real-world systems in which entities interact in non-Euclidean ways [2]. Social networks, citation networks, molecular structures, transportation systems, and knowledge graphs all naturally encode their information as nodes connected by edges. This flexibility has made GNNs a foundational tool in modern machine learning.

Typical downstream tasks include node-level classification (e.g., detecting anomalies or classifying individuals in a network), edge-level prediction (e.g., predicting new friendships or molecular bonds), and graph-level tasks (e.g., predicting molecular properties or detecting fraudulent subgraphs) [1, 3]. Each task requires the model to propagate and refine information across varying neighbourhood radii, which introduces both expressive opportunities and structural challenges, including, most notably, the oversmoothing problem [4].

2 Oversmoothing Problem

A well-documented limitation of deep GNNs is oversmoothing [4, 5], a phenomenon in which node representations become progressively indistinguishable as the number of layers increases. Because each layer aggregates information from local neighbours, deeper architectures repeatedly mix information

across larger neighbourhoods. Past a certain depth, this leads to a homogenization of node embeddings such that the model loses its ability to discriminate between distinct structural roles or feature patterns [6]. Ultimately, oversmoothing degrades task performance, especially on graphs with sparse or heterophilic connectivity [7].

3 Research Question

Our research seeks to evaluate how effectively two prominent techniques, GCNII and DropEdge, mitigate the oversmoothing problem in graph neural networks [8, 5]. We evaluate their performance on heterophilic graph datasets to determine whether methods originally designed for homophilic settings retain their advantage when node similarity assumptions break down.

4 GCNII: Deep Graph Convolution via Initial Residuals and Identity Mapping

To mitigate the oversmoothing problem without sacrificing network depth, we utilize the GCNII architecture. Standard Graph Convolutional Networks (GCNs) suffer from performance degradation as layers are added, primarily because repeated Laplacian smoothing renders node embeddings indistinguishable [1, 4]. We adopt GCNII to enforce a structural memory that preserves node distinctness even in deep networks [5].

The architecture is defined by two primary modifications to the standard propagation rule: **Initial Residual Connections** and **Identity Mapping**.

4.1 Initial Residual Connections

In a standard deep GCN, the input features of the graph are gradually diluted as information propagates through successive layers. To counteract this, GCNII introduces an initial residual connection that links the smoothed representation at the current layer directly back to the original input layer $H^{(0)}$.

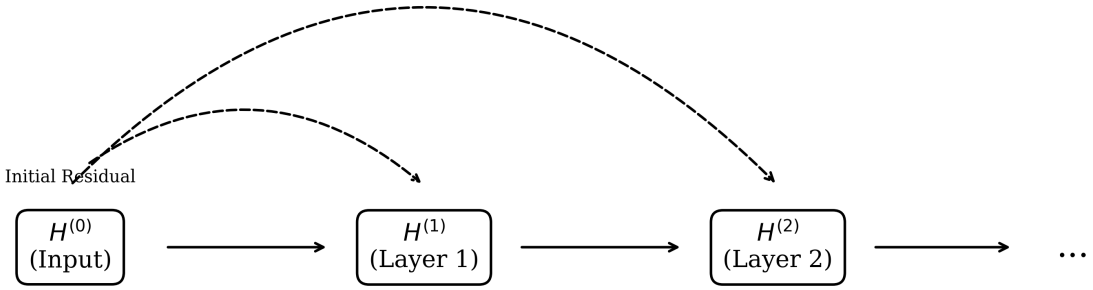

Figure 1: Information flow in GCNII. The initial residual connection acts as a tether, ensuring $H^{(0)}$ contributes to the final embedding regardless of depth [5].

Formally, the update rule for a layer l incorporates the initial feature matrix $H^{(0)}$ via a smoothing coefficient α:

$$H^{(l+1)} = \sigma\left(\left((1-\alpha)\tilde{P}H^{(l)} + \alpha H^{(0)}\right)W^{(l)}\right) \tag{1}$$

This connection acts as a "tether" to the initial node features. Regardless of how many layers L the network possesses—experimentally tested up to 32 layers in our research —the final embedding retains a fraction of the original input signal. This ensures that the local node features are not washed out by the aggregation of neighboring features.

4.2 Identity Mapping

The second component addresses the weight matrix degradation common in deep networks. GCNII employs identity mapping, where an identity matrix I_n is added to the learnable weight matrix W at each layer.

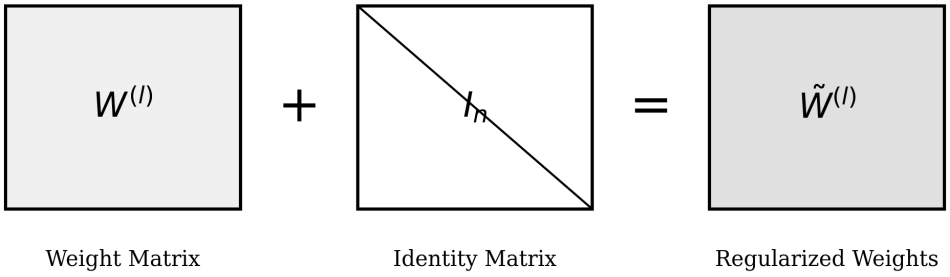

Figure 2: Identity mapping stabilizes the weight matrix $W^{(l)}$, preventing numerical instability and feature collapse in deep layers.

In the context of heterophilic graphs, where neighbours often possess dissimilar labels, this feature is decisive [7]. It allows the network to selectively aggregate information from neighbors while maintaining the unique identity of the central node via the identity matrix I_n.

5 DropEdge: Deep Graph Convolution via Random Edge Removal

DropEdge is a flexible technique designed to facilitate the training of deep GCNs by randomly removing a specific rate of edges from the input graph during each training epoch [8]. By doing so, it acts as a data augmentation tool, generating varied, deformed copies of the original graph to increase data diversity and prevent over-fitting. Simultaneously, it functions as a message-passing reducer; because message passing occurs along edge paths, thinning these connections makes the graph sparser, which is crucial for building deeper architectures. This process is unbiased, as it does not change the expectation of neighbour aggregation after weight normalisation [8]. The fundamental operation of DropEdge is defined by the modification of the graph's adjacency matrix at each training epoch:

$$A_{\text{drop}} = A - A'$$

This equation states that the new adjacency matrix (A_{drop}) is created by taking the original adjacency matrix (A) and subtracting a sparse matrix (A') that contains a random subset of edges.

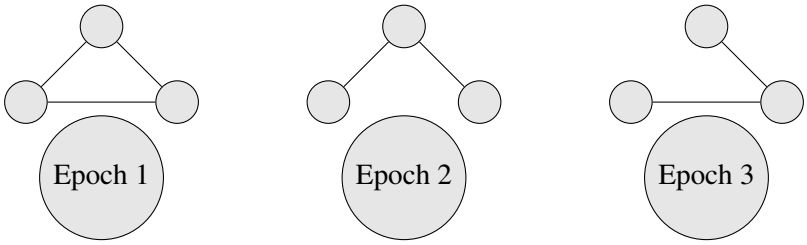

Figure 3: DropEdge samples a different sparsified graph at each training epoch

5.1 DropEdge's Theoretical Mechanisms for Mitigating Oversmoothing

DropEdge mitigates the oversmoothing problem through two primary theoretical mechanisms [8]:

- **Slowing convergence:** It reduces the speed at which node features converge towards a common subspace. Mathematically, it increases the value of the relaxed ε-smoothing layer, allowing the network to be deeper before features become indistinguishable.
- **Reducing information loss:** Information loss is measured by the gap between the original space dimensions and the converging subspace. DropEdge increases the dimension of the converging subspace, thereby preserving more information from the input features as the network deepens.

$$N - \dim(M) > N - \dim(M')$$

N is the total number of nodes, and $\dim(M)$ is the dimension of the subspace the features converge to; a larger gap represents more information loss. When DropEdge removes enough edges to make a connected component disconnected, it increases the dimension of the subspace M. This reduces the gap $(N-M)$, meaning that even if the network is extremely deep, the features retain more information from the original input than they would on the original, more densely connected graph.

6 Methodology

The methodology was designed to compare classification accuracy across training, validation, and test splits, with test accuracy serving as the primary benchmark metric. Comparative evaluation focused on isolating performance differences between GCNII and DropEdge under controlled conditions.

6.1 Dataset utilised

Experiments were conducted on the WebKB benchmark dataset, which consists of the Texas, Cornell, and Wisconsin graphs. These graphs are constructed from university web pages, where nodes correspond to individual pages and edges represent hyperlink relationships between them. Node labels indicate page categories such as student, faculty, course, and project pages.

The WebKB graphs are commonly classified as heterophilic due to the predominance of edges connecting nodes belonging to different classes [7, 9]. This structural characteristic contrasts with homophilic graph settings, in which nodes are more likely to connect to others sharing the same label. In WebKB, intra-class connectivity is sparse, and label similarity among neighboring nodes is low, resulting in weak alignment between node features, local neighborhoods, and class labels.

This heterophilic structure poses a challenging evaluation environment for graph learning models, particularly those relying on neighborhood-based aggregation mechanisms. Consequently, the WebKB dataset has been widely adopted as a benchmark for assessing model robustness under irregular connectivity patterns, structural anomalies, and settings where homophily assumptions do not hold.

6.2 Experimental setup

Both GCNII and DropEdge were evaluated across 2, 4, 8, 16, and 32 layers (with computational limits preventing extensions beyond that). For comparison, the baseline GCN and GraphSAGE models were also implemented to isolate the impact of architectural design on depth scalability [1, 3].

To verify the reliability of implementations, results from the original papers on benchmark citation networks were replicated. Our reproduction closely matched the reported outcomes. Notably, the GCNII authors provide official WebKB implementations for their GCN baselines, lending further confidence to the validity of their comparative claims, whereas DropEdge was primarily benchmarked on homophilic citation graphs.

7 Code

Working of the two sub-methods in the GCNII mitigation technique

```
def forward(self, x, x0, adj):
    """
x:Current␣layer␣node␣features␣(H^l)
x0:Initial␣input␣node␣features␣(H^0)
adj:Normalized␣adjacency␣matrix␣(A_hat)
␣␣␣␣"""

    # 1. GRAPH CONVOLUTION & INITIAL RESIDUAL
    # Smoothes features and anchors them to the initial input features
    support = (1 - self.alpha) * torch.spmm(adj, x) + self.alpha * x0

    # 2. IDENTITY MAPPING
    # Pushes the weight matrix toward Identity to prevent
       over-transformation
    # Beta decreases as depth increases: beta = log(lambda/l + 1)
    output = (1 - self.beta) * support + self.beta * torch.mm(support,
       self.weight)

    return self.activation(output)
```

Working of random edge masking and adjacency re-normalisation logic implemented in DropEdge

```
def apply_dropedge(adj, drop_rate, training=True):
    """
␣␣␣␣adj:␣Adjacency␣matrix␣(A)
␣␣␣␣drop_rate:␣Fraction␣of␣edges␣to␣drop␣(p)
␣␣␣␣training:␣Train␣or␣evaluation␣mode
␣␣␣␣"""

    # Use full graph during validation/testing
    if not training:
        return normalize(adj)

    # 1. RANDOM EDGE MASKING
    # Acts as regularisation by reducing oversmoothing in deep GNNs
    edge_mask = random_bernoulli_mask(adj, keep_prob=1 - drop_rate)
    adj_drop = adj * edge_mask

    # 2. ADD SELF-LOOPS
    adj_drop = adj_drop + identity_matrix(adj_drop.shape[0])

    # 3. ADJACENCY RE-NORMALISATION
    degree = compute_degree(adj_drop)
    D_inv_sqrt = inverse_sqrt_diag(degree)
    adj_norm = D_inv_sqrt @ adj_drop @ D_inv_sqrt

    return adj_norm
```

8 Conclusion

Depth-robust graph neural network architectures were observed to exhibit fundamentally different behavior on heterophilic datasets compared to the homophilic benchmarks on which they were originally

validated. Through evaluation of GCNII and DropEdge across multiple WebKB subsets and increasing network depths, empirical evidence was obtained indicating that oversmoothing emerges earlier and with greater severity in heterophilic settings. As a result, conventional regularization techniques such as DropEdge were found to be substantially less effective. In contrast, the identity mapping and initial residual connections employed by GCNII were shown to preserve discriminative node representations even at increased depth, indicating greater suitability for real-world networks in which relational similarity cannot be assumed.

These findings are significant for two primary reasons. First, a large portion of graph neural network research and downstream applications implicitly assumes homophily, which can lead to misleading assessments of model reliability when applied to heterogeneous or noisy graphs. Second, several high-stakes application domains, including cybersecurity, fraud detection, biological interaction networks, and irregular web-graph analysis, are inherently heterophilic. In such settings, models that suffer from oversmoothing risk suppressing critical anomalies or rare structural patterns.

By demonstrating that architectural design plays a decisive role in depth scalability under heterophilic conditions, these results highlight the necessity of context-sensitive evaluation of graph neural networks rather than reliance on performance reported solely for citation network benchmarks.

8.1 Future work

Future extensions of this research could explore:

- Scaling experiments to 64 and 128 layers to evaluate the true depth limits of GCNII and identify the breaking point of over-smoothing.
- Investigating DropGNN, an alternative approach that removes nodes rather than edges, potentially suppressing noise by pruning inactive or weakly informative nodes [10].
- Incorporating adaptive node and edge pruning strategies into DropEdge to enable application-specific sparsification.
- Applying the models to additional heterophilic datasets (e.g. Actor, Amazon, Chameleon) to assess the generalisability of the observed trends.

References

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

[2] B. Sanchez-Lengeling *et al.*, "A gentle introduction to graph neural networks," *Distill*, 2021.

[3] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[4] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[5] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," *arXiv preprint arXiv:2007.02133*, 2020.

[6] Y. Hou, J. Zhang, J. Cheng, K. Ma, H. Chen, and M.-C. Yang, "Measuring and improving the use of graph information in graph neural networks," *arXiv preprint arXiv:2206.13170*, 2022.

[7] S. Luan, C. Hua, Q. Lu, J. Zhu, L. Ma, L. Wu, X. Wang, M. Xu, X. W. Chang, D. Precup, R. Ying, S. Z. Li, G. Wolf, and S. Jegelka, "The heterophilic graph learning handbook," *arXiv preprint arXiv:2407.09618*, 2024.

[8] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," *arXiv preprint arXiv:1907.10903*, 2019.

[9] K. Wang, G. Zhang, X. Zhang, J. Fang, X. Wu, G. Li, S. Pan, W. Huang, and Y. Liang, "The heterophilic snowflake hypothesis: Training and empowering gnns for heterophilic graphs," *arXiv preprint arXiv:2406.12539*, 2024.

[10] P. A. Papp, K. Martinkus, L. Faber, and R. Wattenhofer, "Dropgnn: Random dropouts increase the expressiveness of graph neural networks," *arXiv preprint arXiv:2111.06283*, 2021.

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2018.

[12] M. Lampert and I. Scholtes, "The self-loop paradox: Investigating the impact of self-loops on graph neural networks," *arXiv preprint arXiv:2312.01721*, 2023.