BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 14/15 pp205–216
DOI: 10.26599/BDMA.2023.9020019
Volume 7, Number 1, March 2024

Limits of Depth: Over-Smoothing and Over-Squashing in GNNs

Aafaq Mohi ud din* and Shaima Qureshi

Abstract: Graph Neural Networks (GNNs) have become a widely used tool for learning and analyzing data on graph structures, largely due to their ability to preserve graph structure and properties via graph representation learning. However, the effect of depth on the performance of GNNs, particularly isotropic and anisotropic models, remains an active area of research. This study presents a comprehensive exploration of the impact of depth on GNNs, with a focus on the phenomena of over-smoothing and the bottleneck effect in deep graph neural networks. Our research investigates the tradeoff between depth and performance, revealing that increasing depth can lead to over-smoothing and a decrease in performance due to the bottleneck effect. We also examine the impact of node degrees on classification accuracy, finding that nodes with low degrees can pose challenges for accurate classification. Our experiments use several benchmark datasets and a range of evaluation metrics to compare isotropic and anisotropic GNNs of varying depths, also explore the scalability of these models. Our findings provide valuable insights into the design of deep GNNs and offer potential avenues for future research to improve their performance.

Key words: Graph Neural Networks (GNNs); learning on graphs; over-smoothing; over-squashing; isotropic-GNNs; anisotropic-GNNs

1 Introduction

The growing use of graph-structured data in a wide range of real-world applications has led to a surge of interest in machine-learning techniques that are specifically designed to operate on graphs. This includes applications such as social networks, recommender systems, hyperlinked web documents, knowledge graphs, and molecule simulation data generated by scientific computation[1–3]. In recent years, Graph Neural Networks (GNNs) have emerged as a powerful method for representation learning on graphs. The effectiveness of GNNs for representation learning on graphs has been demonstrated in various studies[4–6]. One of the unique features of GNNs is their use of neural message passing, which involves exchanging vector messages between nodes and updating them using neural networks. GNNs utilize a message-passing algorithm to update the hidden embedding $h_v^{(k)}$ of each node v in the vertex set V during each iteration. This update is based on the information gathered from the graph neighborhood $N(v)$ of the node. The graph, denoted as $G=(V, E, X)$ consists of a set of vertices V, a set of edges E, and a feature matrix X. During the message-passing process, the algorithm aggregates features from neighboring nodes in $N(v)$, and then the node combines the information gathered from its neighbors with its own information to construct a new vector, and the collection of such new vectors results in the generation of an embedding matrix H. This allows the GNN to effectively capture and leverage the structural information inherent in the graph to improve

• Aafaq Mohi ud din and Shaima Qureshi are with Department of Computer Science and Engineering, National Institute of Technology Srinagar, Srinagar 190006, India. E-mail: aafaq_cs@nitsri.ac.in; shaima@nitsri.net.
* To whom correspondence should be addressed.
Manuscript received: 2023-01-24; revised: 2023-07-10; accepted: 2023-07-18

representation learning. The general GNN framework representing a single message passing layer with adjacency matrix $\tilde{A}$, feature matrix X, and weight matrix W is $H = \tilde{A}^{\mathrm{T}} \cdot X \cdot W^{\mathrm{T}}$ (with a schematic representation of message passing phenomenon given in Fig. 1).

Our approach instantiates a class of isotropic-GNNs based on a node update equation that treats every “edge direction” equally, i.e., each neighbor receives the same weight value, resulting in an equal contribution to the update of the center node. Isotropic-GNNs include vanilla GNNs, such as Graph Convolutional Networks (GCNs)[7] and GraphSage[8]. In contrast, our approach involves implementing anisotropic-GNNs, where each edge direction is treated distinctly in the update equation. This allows for a more nuanced treatment of the structural information present in the graph, potentially leading to improved performance in representation learning tasks. Popular anisotropic-GNN includes Graph Attention Network (GAT)[9]. We have also implemented popular non-GNN approaches, such as Label Propagation (LP)[10] and the Multi-Layer Perceptron (MLP). These methods utilize either label information or the feature matrix, unlike GNNs, which rely on the underlying relationships, as depicted in Fig. 2.

GNNs, however, have the problem that model performance degrades as the number of layers increases. The reason for this is that the deep GNN models lose the node’s local information which is essential for the optimization of a downstream task through many message-passing steps. Over-smoothing[11–13] is the term used to describe this behavior. In addition to the over-smoothing problem, deep graph neural networks can also suffer from over-squashing[14, 15]. Over-squashing occurs when messages are aggregated over a lengthy path, leading to a bottleneck effect where the volume of data is compressed into fixed-size vectors. This can result in information loss and reduced performance for GNNs, particularly in deep architectures where message passing occurs over multiple layers. To put it simply, GNNs can struggle when dealing with a large volume of data that needs to fit into a limited space, and when trying to transmit information over a significant distance. Thus, it is important to consider both the over-smoothing and over-squashing effects when designing deep GNNs to ensure that they can effectively handle the complexity of graph-structured data.

The criterion we use to identify appropriate datasets in our study is their ability to differentiate statistically between the effectiveness of GNNs and other non-GNN models. We collect a total of 5 datasets[16, 17] and 5 models that are widely used in various fields in the PyTorch and PyTorch geometric[18] frameworks. For fair comparisons, we use consistent experimental settings as reported in the literature. To conduct further research, researchers can easily extend our parameters by adding new models with different features and arbitrary datasets from their own experiments. Our work has made the following significant contributions:

- By employing both theoretical and experimental methodologies, our study examines the impact of neighborhood information quality on the accuracy of node classification tasks. We conduct a thorough investigation into the potential over-smoothing

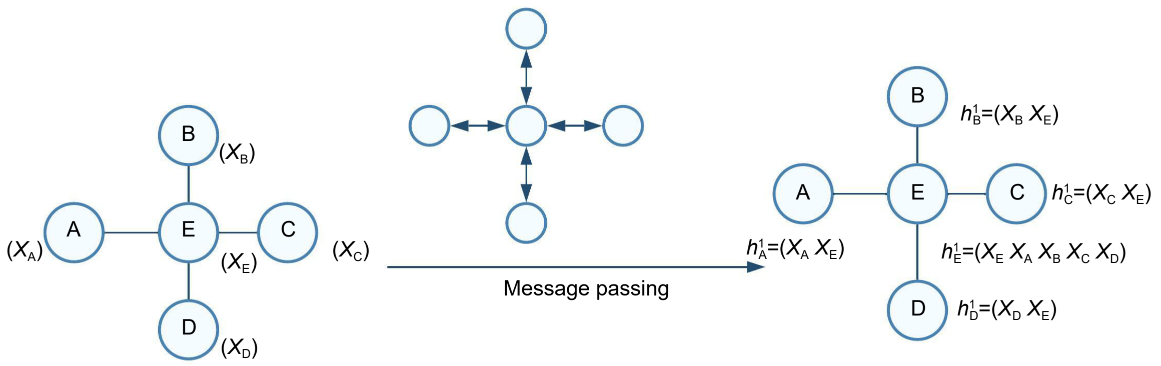


Fig. 1 Conceptual representation of the message passing procedure. Each node starts off with an initial embedding vector (X_A, X_B,…, X_E,) (left). Each pair of linked nodes receives node information via the message-passing method (center). As a consequence, when the message has passed, each node gets its own information as well as neighbor information integrated into a single embedding vector (h^1_A, h^1_B,…, h^1_E) (right). This represents a message passing among 1-hop neighborhoods and this process continues for *k*-hop neighborhoods.

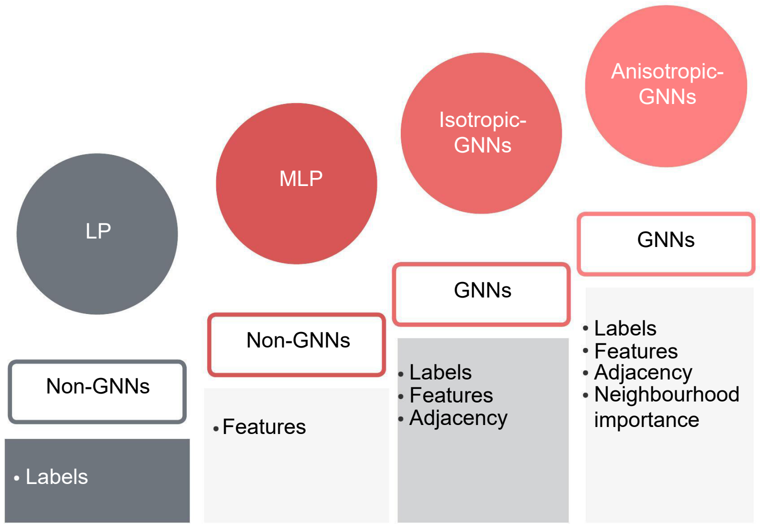


Fig. 2 Highlights of the differences between non-GNN approaches and GNN approaches, specifically isotropic- and anisotropic-GNNs in their use of information for learning tasks on graph-structured data. Non-GNN methods rely solely on features or labels associated with nodes, links, or the entire graph. In contrast, isotropic-GNNs, such as GCN and GraphSAGE incorporate adjacency information to aggregate messages from the local neighborhood up to a certain *k*-hop distance. Anisotropic-GNNs, on the other hand, utilize attention mechanisms to selectively filter and weight incoming messages based on their relevance, allowing for a more fine-grained and targeted message-passing process.

problems that can arise with deep GNN architectures, using datasets of varying sizes: small-scale, medium-scale, and large-scale. We ensure consistent experimental conditions by running each experiment multiple times with the same training settings to eliminate any biases and ensure fair comparisons.

● The study demonstrates the presence of over-squashing issues in deep GNN architectures. As the GNN model incorporates k-hop neighborhood around a node, particularly in large datasets where the average node degree is high, the bottleneck effect is demonstrated. This conclusion is drawn from a comprehensive evaluation of GNN on real-world networks and benchmark datasets of varying sizes.

● We also examine the correlation between node degree and classification accuracy, and our results suggest that nodes with lower degrees pose a greater challenge for achieving high accuracy in classification tasks. Furthermore, we evaluate the scalability of various underlying architectures on multiple datasets and identify the architectures that demonstrate the highest levels of scalability.

The organization of this article can be outlined as follows. Section 2 discusses the related works in the field. Section 3 defines the theoretical and mathematical formulation of several GNN and non-GNN algorithms, and the over-smoothing and over-squashing issues. The experimental benchmark used for this comparative analysis is described in Section 4, along with the parameters chosen to achieve this goal. The results and findings are covered in Sections 5 and 6, respectively. This work ends with the conclusions in Section 7.

2 Related Work

In recent years, GNNs have gained significant attention in the field of machine learning due to their ability to model complex relationships and dependencies in data. This has led to a surge of research efforts focused on developing new architectures, techniques, and applications for GNNs[19–22]. For instance, GNNs have been used for scene graph generation in computer vision tasks, where they have shown significant improvements in performance, as highlighted in a recent survey by Chang et al.[23]

GNNs have also been proposed for hyperspectral image classification, leveraging both spectral and spatial information in the data to improve accuracy. For instance, Hong et al.[24] proposed a GNN-based approach for hyperspectral image classification that uses a graph convolutional neural network to capture the spectral-spatial dependencies in the data. Similarly, in manifold alignment, GNNs have been used to align the manifolds of different modalities and learn a shared representation space, which is a promising approach for land cover and land use classification using GNNs, as demonstrated by Hong et al.[25].

In temporal activity detection, GNNs have been used to model the temporal dependencies between activities and capture common semantics across both seen and unseen activities. For example, Zhang et al.[26] proposed an activity graph transformer that uses label embeddings and graph convolutions to predict activity instances. In spatial-temporal graph modeling, GNNs have been used to capture the interactions among objects in videos, as shown by Li et al.[27]. They proposed a spatial-temporal graph that models object relations in videos and incorporated this information into a shared multilingual visual-semantic embedding space. In zero-shot learning tasks, GNNs have shown promise in learning graph representations for unseen classes. For example, Yan et al.[28] proposed a ZeroNAS technique that uses a GNN to learn a compact graph representation that can generalize to unseen classes in zero-shot learning tasks.

In the context of hyperspectral data analysis, recent studies have demonstrated the potential of GNNs in improving the performance of various tasks. For instance, GNN-based approaches have been proposed to address spectral variability in hyperspectral unmixing[29], and iterative multitask regression frameworks have been presented that dynamically propagate labels on a learnable graph for improved performance in semi-supervised hyperspectral dimensionality reduction[30]. These recent developments highlight the broad potential and versatility of GNNs in hyperspectral data analysis and suggest ongoing efforts to advance this rapidly evolving field. Overall, the growing interest in GNNs in the machine learning community underscores the importance of developing innovative approaches that can leverage the power of graph-based representations for a wide range of applications.

While GNNs have shown great potential in modeling complex data structures, the limits of their depth have not yet been thoroughly explored. Addressing these limits remains an important research challenge, as deeper architectures can potentially capture more complex relationships in data, but may also suffer from over-smoothing and over-squashing, which can lead to decreased performance. To address this issue, this paper presents a comprehensive investigation of the impact of depth on GNNs, studying the phenomena of over-smoothing and over-squashing both theoretically and experimentally. The paper also explores the impact of node degrees on classification accuracy and investigates the scalability of different GNN architectures.

3 Theoretical Background

This section describes the different techniques used in the paper and throws light on the qualitative comparison of the same (as shown in Table 1).

3.1 Non-GNNS

Non-GNNs include a class of models that use either labels associated with nodes (or links or an entire graph) or individual node features but do not consider neighborhood connectivity for a downstream task.

(1) LP

Label propagation is the simplest algorithm for node-level prediction problems that does not involve the use of neural networks. LP is a technique where, given certain unlabeled nodes in the graph, we utilize an iterative method to assign labels to these unlabeled nodes by propagating labels over the dataset using a probabilistic relational classifier[10].

(2) MLP

One of the most fundamental types of neural network model designs is the MLP, in which the unprocessed

Table 1 Embedding equation of GNNs and Non-GNNs.

Category	Model	Embedding equation	Reference
Non-GNN	LP	$P(Y_v = c) = \frac{1}{\sum\limits_{(v,u) \in E} A_{v,u}} \sum\limits_{(v,u) \in E} A_{v,u} \cdot P(Y_v = c)$	[10]
	MLP	$H = a\,(b + W \cdot X)$	[31]
	Random forests+	$\hat{C}_{\text{rf}}^{B}(x) = \text{majority vote } \left\{\hat{C}_b(x)\right\}_1^B$	[32]
GNN	GCN*	$H = \tilde{D}^{-1/2} \cdot \tilde{A}^{\mathrm{T}} \cdot \tilde{D}^{-1/2} \cdot X \cdot W^{\mathrm{T}}$	[7]
	GraphSAGE*	$H = W_1 \cdot X + W_2 \cdot \text{mean}\,(\tilde{A}^{\mathrm{T}} \cdot X)$	[8]
	GAT**	$H = \tilde{A}^{\mathrm{T}} \cdot W_{\text{alpha}} \cdot X \cdot W^{\mathrm{T}}$	[9]

Note: * denotes isotropic-GNNs and ** denotes anisotropic-GNN.

node input characteristics are "fed-forward" through a number of layers of computation to generate an output that corresponds to a probability distribution across the predicted classes. Because of the non-linearity introduced by the activation functions in the layers, our neural network can learn more complicated functions. These activation functions are helpful for deeper GNNs as well as MLPs[31]. ReLU activation functions are often used in each hidden layer of MLP models, while a Softmax activation function is present in the output layer. An MLP analyses nodes in this manner, but it is less accurate since it cannot take the adjacency matrix into account.

Table 1 shows that for a given node v in a graph, the associated label is represented by the symbol c. The adjacency matrix, represented by the symbol $\tilde{A}$, and the feature matrix, represented by X, are also associated with the graph nodes. The weight matrix is represented by W, while the final embedding matrix generated is represented by H. In addition to this, the bias is represented by the symbol b, and the activation function is represented by a[32].

3.2 GNNs

In the context of GNNs, a GNN layer can be conceptualized as a step in the message-passing process, where each node adjusts its state by gathering messages from its immediate neighbors. The key distinguishing factor between various GNN types is the manner in which each node blends its own representation with that of its neighbors. Figure 1 illustrates the fundamental message-passing process. However, in most applications, it is also necessary to enable interactions among nodes that are not directly linked, and this is accomplished by incorporating multiple layers of GNNs.

(1) Isotropic aggregation and derived models.

In isotropic-GNNs, every neighbor has the same importance while passing messages to the central node[7, 8]. No weighting factor is assigned to nodes instead an adjacency matrix is applied to the feature matrix to generate the resulting embedding matrix without taking into account the attention matrix. The general embedding equation for isotropic-GNNs simply has an adjacency matrix $\tilde{A}$, a feature matrix X associated with a node/link or an entire graph, and a weight matrix W,

$$H = \tilde{A}^{\mathrm{T}} \cdot X \cdot W^{\mathrm{T}} \tag{1}$$

(2) Anisotropic aggregation and derived models.

Anisotropic-GNNs are based on the simple idea that some nodes are more important than others regardless of their node degrees. In order to account for the significance of each neighbor, an attention mechanism has been introduced (originally proposed in Ref. [9]) which assigns weightage to each connection in the network. So instead of calculating static weights based on node degrees like isotropic-GNNS, they assign dynamic weights to node features through a process called self-attention. Attention-based GNN models typically use non-negative attention weights to discount the contribution from dissimilar neighbors. The mechanism assigns a weighting factor (attention score) to each connection involving weight matrix W_{alpha} that contains the final attention coefficient of all the nodes,

$$H = \tilde{A}^{\mathrm{T}} \cdot W_{\mathrm{alpha}} \cdot X \cdot W^{\mathrm{T}} \tag{2}$$

3.3 Over-smoothing

The problem of over-smoothing in GNNs was first identified in Ref. [11], where it was shown that when Laplacian smoothing is repeatedly applied, the attributes of nodes within each connected component of the network tend to converge to a uniform value. This is because deep GNN models go through a lot of message-passing steps and lose the local knowledge of the node, which is crucial for excellent model performance. As a result, GNN performance suffers when there are several layers stacked. The problem of over-smoothing in GNNs restricts the ability of these models to become deep, that is, to have a large number of message-passing steps, which is crucial for capturing the overall structural characteristics of the graph. This issue is clearly illustrated in Fig. 3[33].

3.4 Over-squashing

In order for a GNN to incorporate information from nodes that are k-hops away, it should have at least K layers; otherwise, there is a risk of under-reaching, where distant nodes remain unaware of one another. It is important to note that tasks that require long-range interaction would require as many GNN layers as the interaction range to avoid under-reaching. With each additional layer, the receptive area of a node expands exponentially. Nevertheless, this growth causes the information from the enlarged receptive field to be squeezed into fixed-length node vectors, resulting in over-squashing, as discussed in Refs. [14, 34].

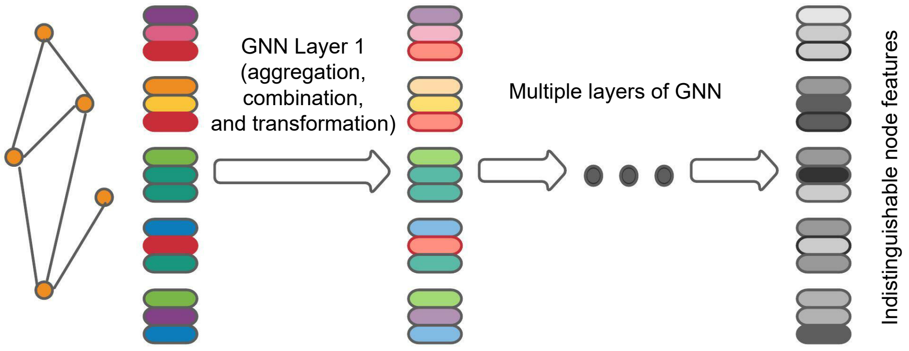


Fig. 3 Visual representation highlighting the challenge of performing downstream tasks when the feature vectors of all nodes become indistinguishable due to over-smoothing, which occurs when the number of layers in a GNN increases to the extent that all the feature vectors become identical. As depicted, the feature vectors are initially distinguishable prior to undergoing the GNN layer, but after a certain number of layers, they lose their distinctiveness. An inherent issue with GNNs is over-smoothing, which is caused by the progressive loss of node-specific information over multiple cycles of GNN message forwarding. This results in the information gathered from nearby nodes after each GNN layer becoming more prominent in the new node representations, leading to the loss of unique node information.

Consequently, the network is capable of learning only short-range signals from the training data and is unable to transmit information originating from distant nodes, as clearly depicted in Fig. 4[33].

4 Experimental Case-Study

4.1 Unified view

This paper focuses on experimentation and analysis to gain a statistical understanding of the inherent limitations of deep graph neural networks. The aim is to establish a firm foundation for future research advancements in the field. The primary objective is to explore and address fundamental research questions related to this topic.

RQ1: Can the accuracy of classification be affected by the quality of neighborhood information? **RQ2**: Are deep anisotropic and isotropic GNNs susceptible to over-smoothing? **RQ3**: Can anisotropic-GNNs serve as a solution to the problem of over-squashing in deep GNNs and alleviate bottlenecks? **RQ4**: Can the degree of a node affect the accuracy of classification? **RQ5**: Can anisotropic and isotropic GNN be scaled up to handle large-scale datasets?

To this end, we conduct experiments using a different range of node classification datasets.

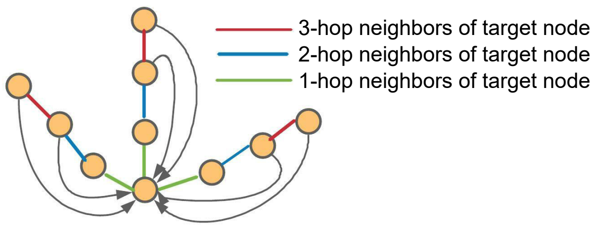

3-hop neighbors of target node
2-hop neighbors of target node
1-hop neighbors of target node

Fig. 4 Visual representation depicting the phenomenon of information overflow that occurs when a GNN becomes deeper and begins to incorporate information from the k-hop neighborhoods surrounding a target node. As the amount of data being processed grows exponentially, it gets condensed into fixed-size vectors due to a bottleneck. This can lead to the compression of critical information, resulting in the loss of important details. This bottleneck phenomenon can be observed in various GNN architectures and can severely impact the network's performance, as depicted in this figure.

4.2 Numerical datasets

The effectiveness of the proposed models is assessed using three well-known citation network datasets, namely Cora[35], Citeseer[36], and PubMed[37], which are benchmark citation network datasets used for evaluating graph-based machine learning models. These datasets consist of scientific publications that are categorized into multiple topics and represented as citation networks, where each publication is connected to others through citation links. Each publication is represented as a sparse bag-of-words vector, with binary values indicating the presence or absence of each word in the publication. In addition, we evaluate the effectiveness of our models using the OGBN-Arxiv dataset[17], which is a large-scale citation network dataset containing over 1.4 million scientific publications, along with their citation links, abstracts, and authors and can be used to determine a technique's scalability due to its large number of nodes and edges. We carefully design the prediction task and data split for each dataset to ensure that achieving strong prediction performance would have a significant impact on the associated application. Table 2 provides an overview of these datasets. This approach combines academic rigor with creative thinking to evaluate the models' effectiveness in various contexts.

4.3 Experimental setup

The present study aims to identify suitable datasets of varying sizes (small, medium, and large) that possess the ability to distinguish the performance of GNNs through statistical analysis. PyTorch Geometric[18] is a widely-used Python package for developing graph-based deep learning models. It offers preprocessed versions of benchmark datasets, including the Cora, Citeseer, PubMed, and OGBN-Arxiv citation network datasets, that can be easily used with popular deep learning frameworks like PyTorch. This makes it a convenient tool for researchers and practitioners working on graph-based machine-learning problems. All the experiments are run on Quadro P4000 with 64 GB GPU memory architecture with predefined optimizer and learning rates while keeping all other parameters fixed using the PyTorch and PyTorch geometric frameworks. For each setting, we run 10 times and report the average results.

5 Result

RQ1: Impact of quality neighborhood information on classification accuracy (see Table 4 and Fig. 5).

RQ2: Isotropic- and anisotropic-GNNs for node classification task under different depths (see Table 3 and Figs. 6 and 7).

RQ3: Over-squashing effect in GNNs. No such techniques have yet been presented in the literature that would quantify over-squashing, despite the fact that it has lately been documented in the literature. We will quantify the problem with the help of a graph showing performance drop as we increase the number of hops (h) during the training phase on a large dataset (see Fig. 8).

RQ4: Average degree per node versus classification accuracy (see Fig. 9).

RQ5: Training time computation (see Fig. 10).

6 Finding

Here is the list of observations and findings from the given experiments.

F1: Our experimental analysis suggests that the accuracy of classification is influenced by the quality of information gathered from neighboring nodes, as illustrated in Fig. 5 and Table 3. Isotropic models aggregate information indiscriminately from all neighboring nodes, while anisotropic models leverage attention scores to minimize the impact of irrelevant neighbor information. Non-GNN models, which do not consider neighbor information, perform poorly on node classification tasks.

F2: Anisotropic models offer a degree of relief from the over-smoothing problem since they can selectively focus on specific nodes during message passing, utilizing non-negative attention weights to reduce the impact of dissimilar neighbors. As demonstrated in Fig. 6, the rate at which performance deteriorates in anisotropic models is lower than that in isotropic models, as confirmed by our experimental results in Table 3.

F3: Our findings illustrate that GNNs that distribute incoming edges evenly, such as isotropic-GNNs, are considerably more vulnerable to the problem of over-squashing than anisotropic-GNNs. This is clearly demonstrated in Fig. 7, where the GCN exhibits underfitting of the training data after four hops, whereas the GAT shows this phenomenon after six hops.

F4: We argue that poorly connected nodes might negatively impact performance. The classification accuracy of Citeseer is mostly low on all GNN models as shown in Table 3. Here, we determine the correctness of the GCN model for each degree on Citeseer and

Table 2 Statistical characteristics of the node classification datasets.

Dataset	Number of nodes	Number of features	Number of edges	Number of classes	Number of average node degrees	Number of nodes for training/validation/testing
Cora{*}{+}	2708	1433	5429	7	3.90	140/500/1000
Citeseer{*}{+}	3327	33 703	4732	6	2.77	120/500/1000
Pubmed{**}{+}	19 717	500	44 338	3	4.50	60/500/1000
OGBN-Arxiv{***}{+}	169 343	128	1 166 243	40	13.70	91/30 000/47 000
OGBN-Products{***}{+}	2 449 029	100	61 859 140	47	50.50	196 000/49 000/2 204 000

Note: + denotes citation network datasets; * denotes small-scale datasets; ** denotes medium-scale datasets; and *** denotes large-scale datasets.

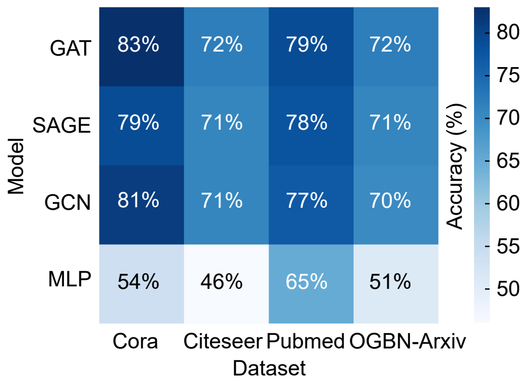


Fig. 5 Heatmap showing the accuracy of models under consideration in Table 3. The heatmap colors represent the accuracy of each model on each dataset, with warmer colors indicating higher accuracy. The annotations within the heatmap show the accuracy scores of MLP, GCN, GraphSAGE, and GAT on the Cora, Citeseer, PubMed, and OGBN-Arxiv datasets. MLP achieves the lowest accuracy overall as no information is aggregated from neighbors. GCN achieves higher accuracy than MLP as information is aggregated from *k*-hop neighborhood around a node. GAT achieves the highest overall accuracy, with the highest accuracy on all three datasets. These results indicate that GAT makes each node more connected to similar nodes than dissimilar nodes by applying attention scores during message passing.

Pubmed datasets as shown in Fig. 8. These findings support our hypothesis that nodes with few neighbors are actually more difficult to categorize. This is because GNNs are designed in such a way that you can aggregate more information the more relevant connections you have.

F5: Attention-based models, due to the complexity of computing attention scores, are not scalable to handle large datasets within a reasonable time frame. In contrast, isotropic models are faster than anisotropic models, as demonstrated in Fig. 9, across all datasets. GraphSAGE[8], on the other hand, is a highly efficient architecture that can handle large graphs. While it may not be as accurate as GCN or GAT, it is a crucial model for managing massive amounts of data. This impressive speed is achieved through a smart blend of neighbor sampling and quick aggregation techniques.

Based on the above findings a qualitative comparison of the underlying techniques is done in Table 5.

7 Conclusion

In conclusion, our study highlights the tradeoff between depth and expressiveness in GNN architectures and the need for more effective models that can propagate relevant information while incorporating both local and global neighborhoods without over-smoothing or over-squashing the data. The use of attention models and taking node degrees into account can improve classification accuracy, but they also introduce computational challenges.

Our findings provide valuable insights into the impact of depth on GNNs and shed light on the phenomena of over-smoothing and over-squashing. These findings can inform the development of more effective GNN architectures for hyperspectral data analysis and other applications. The ongoing efforts to develop such architectures can help address the

Table 3 Results in terms of classification accuracy for various datasets under various depth (or layers, 1, 2, 4, 8, 16, and 32) with mean accuracy ± across various data splits. We also provide the layer at which the particular model performs best (highlighted in bold), broken down by dataset and GNN model.

(%)

Model	Dataset	1 layer	2 layer	4 layer	8 layer	16 layer	32 layer
GCN	Cora	55.0 ± 0.40	**81.30** ± 0.40	77.48 ± 0.40	64.78 ± 0.20	48.55 ± 0.45	32.76 ± 0.14
	Citeseer	48.4 ± 1.60	**71.10** ± 0.70	63.57 ± 1.40	34.20 ± 1.50	33.32 ± 0.45	28.25 ± 2.32
	PubMed	53.7 ± 0.40	**77.80** ± 0.40	72.88 ± 0.44	62.25 ± 7.32	55.50 ± 2.45	42.76 ± 3.14
	OGBN-Arxiv	56.0 ± 5.40	**70.06** ± 0.19	70.48 ± 0.62	66.78 ± 0.20	56.55 ± 0.45	52.76 ± 6.60
GraphSAGE	Cora	52.05 ± 0.45	**79.08**±1.40	77.52 ± 1.75	70.32 ± 1.20	61.55 ± 0.45	42.76 ± 1.14
	Citeseer	54.66 ± 2.40	**71.22**±1.40	65.48 ± 2.30	49.78 ± 0.25	30.72 ± 5.45	28.76 ± 2.30
	PubMed	56.25 ± 1.60	**78.22**±1.66	72.40 ± 2.40	69.78 ± 1.20	56.45 ± 2.45	42.16 ± 2.26
	OGBN-Arxiv	52.06 ± 6.40	**71.22**±2.40	69.48 ± 2.00	58.72 ± 2.25	52.70 ± 1.45	43.26 ± 3.14
GAT	Cora	58.00 ± 0.20	**83.50**±0.50	77.48 ± 2.40	70.78 ± 2.20	62.65 ± 2.45	59.76 ± 2.14
	Citeseer	50.00 ± 3.40	**71.80**±0.40	67.52 ± 1.40	61.78 ± 1.20	55.55 ± 3.45	52.76 ± 0.14
	PubMed	58.02 ± 1.40	**79.42**±0.50	75.48 ± 2.40	70.78 ± 1.20	62.55 ± 2.45	57.26 ± 3.75
	OGBN-Arxiv	59.00 ± 0.40	**71.88**±0.44	68.48 ± 0.05	60.78 ± 2.20	52.55 ± 1.45	48.26 ± 0.06

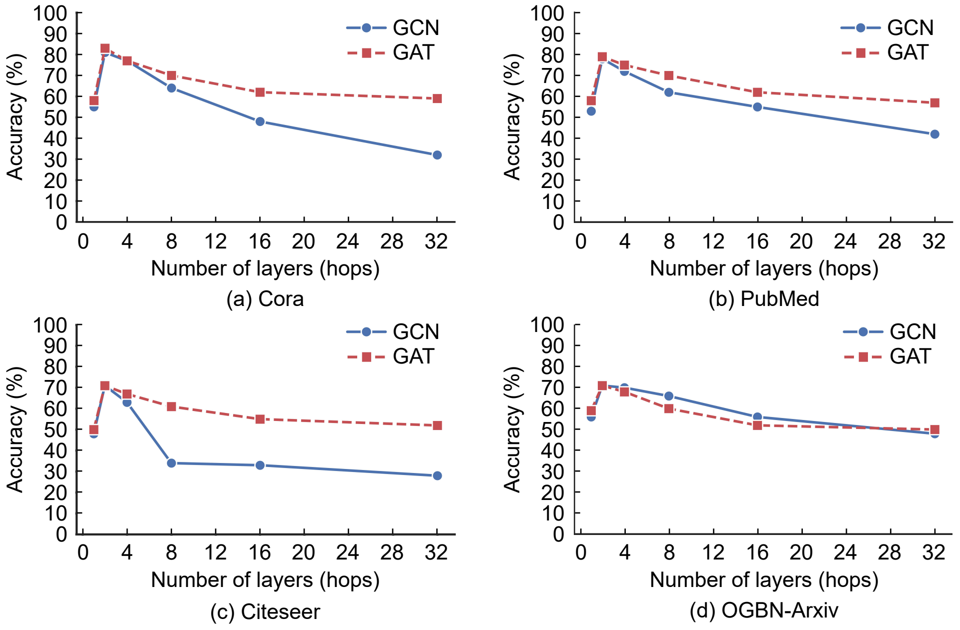


Fig. 6 Plot showing that increasing the number of layers or hops in the GNN beyond a certain point may not lead to a corresponding increase in accuracy but instead a decrease. GCN is more prone to over-smoothing as compared to GAT as shown on different datasets.

Table 4 Results of classification accuracy for various datasets utilizing non-GNNs label propagation technique, and multilayer perceptron are summarised.

(%)

Model	Dataset	Accuracy
LP	Cora	54.50
	Citeseer	44.36
	PubMed	64.12
	OGBN-Arxiv	65.25
MLP	Cora	55.25
	Citeseer	46.50
	PubMed	65.20
	OGBN-Arxiv	51.60

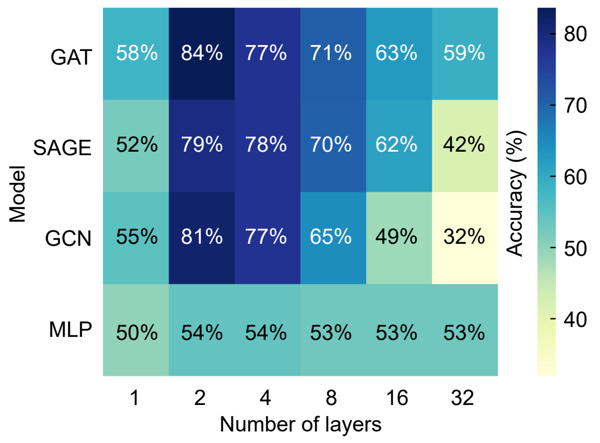


Fig. 7 Heatmap showing the accuracy of various GNN and non-GNN models with increasing depth (number of layers). As the depth increases, the accuracy of the models decreases, indicating the presence of over-smoothing.

tradeoff of depth and improve the performance of GNNs in a variety of tasks.

Future research can investigate the impact of different types of variability on the performance of GNN architectures to improve their robustness and generalization in real-world applications. In addition, the combination of GNNs with other machine learning techniques, such as zero-shot learning and transfer learning, can improve the accuracy and generalization of graph-based models.

Overall, the growing interest in GNNs and their potential for modeling complex data structures underscores the importance of advancing this rapidly evolving field. By developing innovative approaches that can leverage the power of graph-based representations while addressing the limitations of current approaches, we can continue to make significant progress in machine learning and improve the accuracy and interpretability of data analysis across domains.

References

[1] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, doi:10.1016/j.aiopen.2021.01.001.

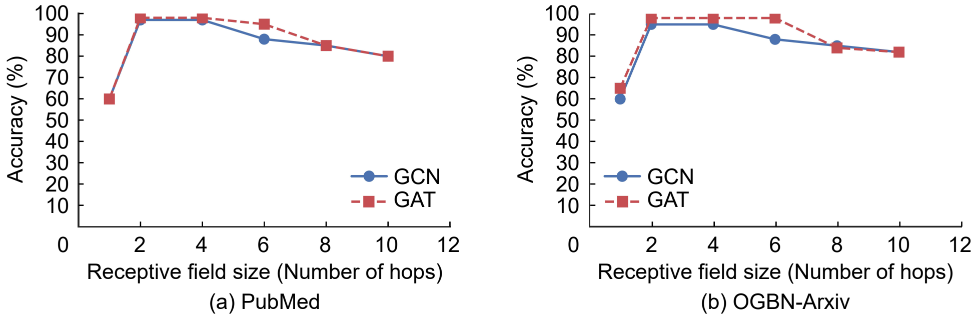


Fig. 8 Comparison of the training accuracy of GCN and GAT models on two large datasets, Pubmed and OGBN-Arxiv, as the number of hops (receptive field) increases. Our results reveal that the training accuracy of the GCN model decreases after four hops, while that of the GAT model decreases after six hops. It is evident that over-squashing has a more pronounced impact on the GCN model than on the GAT model.

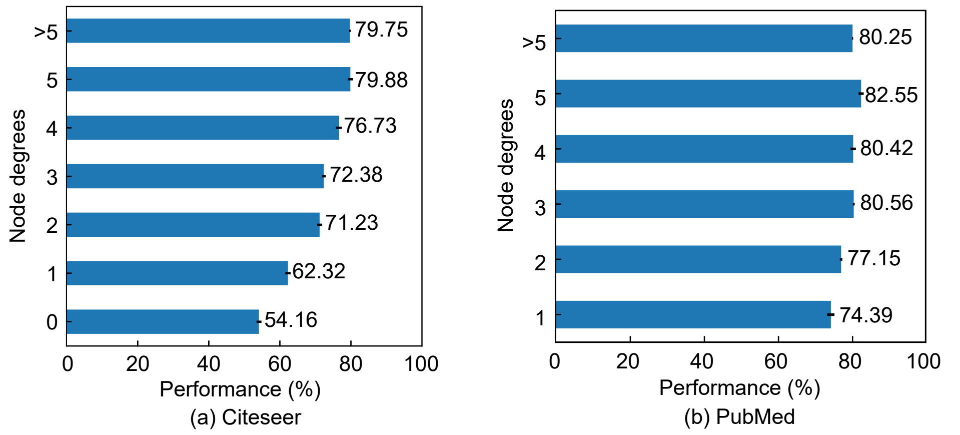


Fig. 9 Uncovering critical role of node degrees in GNNs. Our investigation into the impact of node degree on classification accuracy has uncovered that nodes with fewer neighbors are more challenging to classify accurately. Specifically, on the Citeseer dataset, we observe that isolated nodes exhibit the lowest classification accuracy.

Table 5 Qualitative comparison among isotropic-GNNs, anisotropic-GNNs, and Non-GNNs.

Method	Attention matrix (W_{att})	Training time	Accuracy	Over-smoothing	Over-squashing
Isotropic-GNNs	Not applied	Low	Low	More prone	More prone
Anisotropic-GNNs	Applied	Very high	High	Less prone	Less prone
Non-GNNs	Not applied	Medium	Very low	–	–

[2] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu, Graph learning: A survey, *IEEE Trans. Artif. Intell.*, vol. 2, no. 2, pp. 109–127, 2021.

[3] Y. Rong, T. Xu, J. Huang, W. Huang, H. Cheng, Y. Ma, Y. Wang, T. Derr, L. Wu, and T. Ma, Deep graph learning: Foundations, advances and applications, in *Proc. 26th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, https: //doi.org/10.1145/3394486.3406474 2020.

[4] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, Geometric deep learning: Going beyond Euclidean data, *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, 2017.

[5] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, Neural message passing for quantum chemistry, in *Proc. 34th Int. Conf. Machine Learning*, Sydney, Australia, 2017, pp. 1263–1272.

[6] Z. Liu and J. Zhou, *Introduction to Graph Neural Networks*. Cham: Switzerland, Springer, 2020.

[7] T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, in *Proc. 5th Int. Conf. Learning Representations*, doi:10.48550/arXiv.1609.02907.

[8] W. L. Hamilton, R. Ying, and J. Leskovec, Inductive representation learning on large graphs, in *Proc. 31st Int. Conf. Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 1025–1035.

[9] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, Graph attention networks, arXiv preprint arXiv: 1710.10903, 2017.

[10] F. Wang and C. Zhang, Label propagation through linear neighborhoods, in *Proc. 23rd Int. Conf. Machine Learning*,

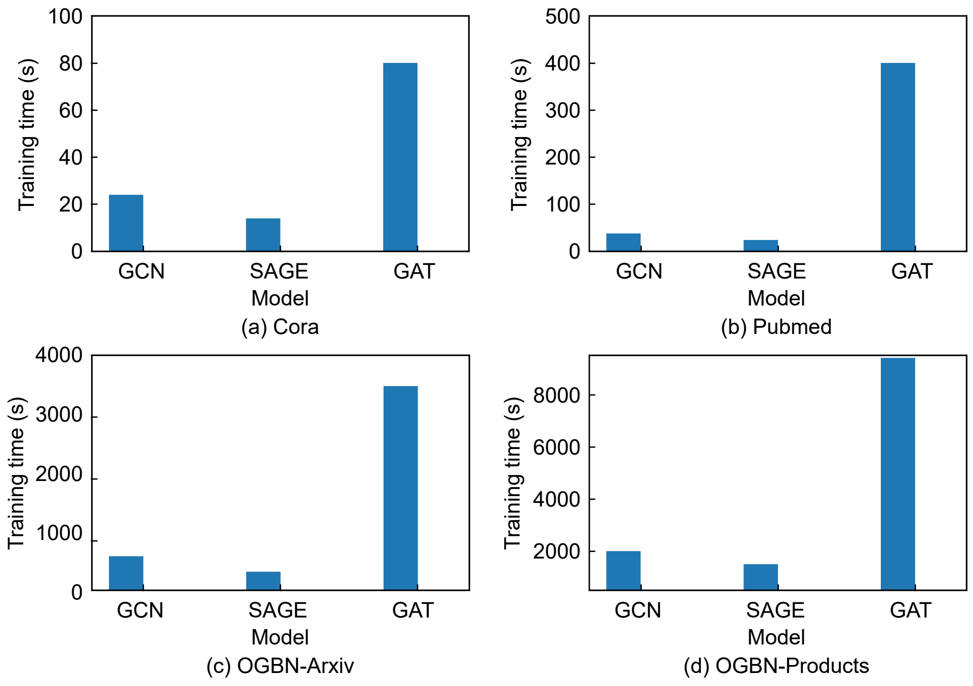


Fig. 10 Assessment of training time requirements of three GNN model, GCN, GraphSAGE, and GAT, on various datasets. It is observed that GAT has a significantly higher training time compared to GCN, which can be attributed to the complex nature of calculating attention scores for each edge. Additionally, we noted that the training time for anisotropic models increases exponentially as the number of layers increases, in contrast to isotropic models.

Pittsburgh, PA, USA, 2006, pp. 985–992.

[11] Q. Li, Z. Han, and X. M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in *Proc. 32nd AAAI Conf. Artificial Intelligence*, New Orleans, LA, USA, 2018, pp. 3538–3545.

[12] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, Measuring and relieving the over-smoothing problem for graph neural networks from the topological view, in *Proc. 34th AAAI Conf. Artificial Intelligence*, New York, NY, USA, 2020, pp. 3438–3445.

[13] L. P. Xhonneux, M. Qu, and J. Tang, Continuous graph neural networks, in *Proc. 37th Int. Conf. Machine Learning*, doi:10.48550/arXiv.1912.00967.

[14] U. Alon and E. Yahav, On the bottleneck of graph neural networks and its practical implications, https: //doi.org/ 10.48550/arXiv.2006.05205, 2020.

[15] D. Lukovnikov and A. Fischer, Improving breadth-wise backpropagation in graph neural networks helps learning long-range dependencies, http: //proceedings.mlr.press/ v139/lukovnikov21a/lukovnikov21a.pdf, 2021.

[16] Z. Yang, W. Cohen, and R. Salakhudinov, Revisiting semi-supervised learning with graph embeddings, in *Proc. 33rd Int. Conf. Machine Learning*, New York, NY, USA, 2016, pp. 40–48.

[17] W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong, and J. Leskovec, OGB-LSC: A large-scale challenge for machine learning on graphs, doi:10.48550/arXiv.2103.09430.

[18] M. Fey and J. E. Lenssen, Fast graph representation learning with PyTorch geometric, arXiv preprint arXiv: 1903.02428, 2019.

[19] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, A comprehensive survey on graph neural networks, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, 2021.

[20] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, Graph neural networks: A review of methods and applications, *AI Open*, vol. 1, pp. 57–81, 2020.

[21] Z. Zhang, P. Cui, and W. Zhu, Deep learning on graphs: A survey, *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 249–270, 2022.

[22] X. Guo and L. Zhao, A systematic survey on deep generative models for graph generation, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5370–5390, 2022.

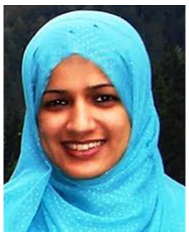
[23] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann, A comprehensive survey of scene graphs: Generation and application, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 1–26, 2023.

[24] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot, Graph convolutional networks for hyperspectral image classification, *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 7, pp. 5966–5978, 2021.

[25] D. Hong, N. Yokoya, N. Ge, J. Chanussot, and X. X. Zhu, Learnable manifold alignment (LeMA): A semi-supervised cross-modality learning framework for land cover and land use classification, *ISPRS J. Photogramm. Remote Sens.*, vol. 147, pp. 193–205, 2019.

[26] L. Zhang, X. Chang, J. Liu, M. Luo, Z. Li, L. Yao, and A. Hauptmann, TN-ZSTAD: Transferable network for zero-

shot temporal activity detection, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3848–3861, 2023.
[27] M. Li, P. Y. Huang, X. Chang, J. Hu, Y. Yang, and A. Hauptmann, Video pivoting unsupervised multi-modal machine translation, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3918–3932, 2023.
[28] C. Yan, X. Chang, Z. Li, W. Guan, Z. Ge, L. Zhu, and Q. Zheng, ZeroNAS: Differentiable generative adversarial networks search for zero-shot learning, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 9733–9740, 2022.
[29] D. Hong, N. Yokoya, J. Chanussot, and X. X. Zhu, An augmented linear mixing model to address spectral variability for hyperspectral unmixing, *IEEE Trans. Image Process.*, vol. 28, no. 4, pp. 1923–1938, 2019.
[30] D. Hong, N. Yokoya, J. Chanussot, J. Xu, and X. X. Zhu, Learning to propagate labels on graphs: An iterative multitask regression framework for semi-supervised hyperspectral dimensionality reduction, *ISPRS J. Photogramm. Remote Sens.*, vol. 158, pp. 35–49, 2019.
[31] Y. Hu, H. You, Z. Wang, Z. Wang, E. Zhou, and Y. Gao, Graph-MLP: Node classification without message passing in graph, arXiv preprint arXiv: 2106.04051, 2021.
[32] L. Breiman, Random forests, *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[33] A. Mohi ud din and S. Qureshi, A review of challenges and solutions in the design and implementation of deep graph neural networks, *Int. J. Comput. Appl.*, vol. 45, no. 3, pp. 221–230, 2023.
[34] P. Barceló, E. V. Kostylev, M. Monet, J. Pérez, J. Reutter, and J. P. Silva, The logical expressiveness of graph neural networks, http: //grlearning.github.io/papers/92.pdf, 2020.
[35] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, Automating the construction of internet portals with machine learning, *Inf. Retr.*, vol. 3, no. 2, pp. 127–163, 2000.
[36] C. L. Giles, K. D. Bollacker, and S. Lawrence, CiteSeer: An automatic citation indexing system, in *Proc. 3rd ACM Conf. Digital Libraries*, Pittsburgh, PA, USA, 1998, pp. 89–98.
[37] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, Collective classification in network data, *AI Mag.*, vol. 29, no. 3, pp. 93–106, 2008.



Shaima Qureshi is an associate professor at Department of Computer Science and Engineering, National Institute of Technology Srinagar, India. She received the BEng degree from BITS Pilani, India in 2004, the MEng degree in computer science from Syracuse University, USA in 2006, and the PhD degree from National Institute of Technology Srinagar, India in 2018. She joined National Institute of Technology Srinagar in 2008. Prior to joining the academic field, she worked as a senior quality assurance engineer for two years in the software industry in USA and India. Apart from teaching and administrative duties, she is guiding PhD students. She has many publications to her credit. Her areas of research include mobile networks, algorithms, machine learning, and artificial intelligence.

Aafaq Mohi ud din received the BEng and MEng degrees in computer science and engineering from University of Kashmir, Srinagar, India. He is currently a PhD candidate in geometric deep learning at Department of Computer Science and Engineering, National Institute of Technology Srinagar (NIT), Srinagar, India. Prior to that, he worked as a faculty at NIT Srinagar, India. His research interests include geometric deep learning, graph learning, and reliability.