Enhancing Message Passing Neural Networks with Selective Long-Range Connections to Mitigate Over-Squashing

Maximilian Seeliger
Institute of Information Systems Engineering
Technical University of Vienna
Vienna, AUT
maximilian.seeliger@tuwien.ac.at

Abstract

This paper introduces a novel approach to mitigate over-squashing in Message Passing Neural Networks (MPNNs) by establishing selective long-range connections in the network's final layer. Over-squashing, a challenge in processing large-scale graph data, limits MPNNs' ability to handle extensive graph structures, impacting fields such as molecular biology and social network analysis. Our method strategically connects distant nodes within the graph topology, allowing for efficient information flow over extended distances. This selective connectivity balances global information flow and local feature distinctiveness, enhancing MPNNs' applicability and accuracy in complex scenarios. The approach is tested using Graph Convolutional Networks (GCN) and Graph Isomorphism Networks (GIN) on the Tree-NeighborsMatch dataset, demonstrating significant improvements in model performance and providing insights into designing efficient MPNN architectures.

1 Introduction

Graph neural networks (GNNs) have emerged as a powerful tool in machine learning for tasks involving graph-structured data. Their relevance extends across a wide range of applications, from social network analysis to drug discovery in chemistry [Duvenaud et al., 2015, Tan et al., 2019, Fan et al., 2019, Wu et al., 2020]. Central to the success of GNNs is their ability to leverage the rich relational information inherent in graphs, enabling them to learn complex patterns that traditional neural network architectures struggle with.

At the heart of many GNN architectures is the message passing neural network (MPNN) framework. Introduced by Gilmer et al. [2017], MPNNs provide a general model for iteratively updating node representations by aggregating information from their neighbors. This message passing approach effectively captures local graph structures. For an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, each node $v \in V$ has an associated feature vector x_v, and each edge $(u, v) \in E$ may also have an associated feature vector e_{ev}. At each iteration t, a message function M_t is used to compute a message m_v^t for each node v based on the features of the node, the features of its neighbors $N(v)$ and potentially also the features of the edges connecting them. Formally,

$$m_v^{t+1} = \sum_{u \in N(v)} M_t(h_v^t, h_u^t, e_{vu}) \tag{1}$$

where h_v^t is the hidden state of node v at step t. Initially, the hidden states of each node v are derived from their feature vector x_v.

Consequently, the hidden state is updated at each iteration using an update function U_t which combines the message m_v^t with the previous hidden state h_v^t:

$$h_v^{t+1} = U_t(h_v^t, m_v^t) \tag{2}$$

After all T message passing iterations, a readout operation R is performed to compute final representations:

$$\hat{y} = R(\{h_v^T : v \in V\}) \tag{3}$$

The functions M_t, U_t and R are learnable and often implemented using neural network layers [Gilmer et al., 2017].

This framework, although powerful, has an obvious drawback. The reduction of sizeable neighborhood messages and aggregation with the hidden state results in information being lost due to the limited expressive power of the fixed size vector used to represent the next hidden state. For tasks, where messages have to be passed along k intermediate nodes, the problem increases with the fast-growing k-hop neighborhood. This challenge is called *Over-Squashing* [Alon and Yahav, 2021, Topping et al., 2021].

The number of layers in the GNN, namely its depth, is a primary factor in the analysis of Over-Squashing. However, it also effects two similar phenomena: Over-Smoothing and Under-reaching.

Over-smoothing occurs as a result of repeated application of the aggregation function in deep GNNs. This process is akin to smoothing out the features across the graph, leading to a loss of useful information. This results in a homogenization of node features, where distinct nodes in the graph tend to have similar representations. It significantly impairs the model's ability to capture and utilize the intricate structural and feature-based nuances present in the graph, thereby reducing the model's overall performance [Oono and Suzuki, 2019, Cai and Wang, 2020, Rusch et al., 2023].

Under-Reaching, on the other hand, is the GNNs failure to capture sufficient contextual information from the graph. It typically occurs in shallow GNNs with a limited number of layers. In such models, the receptive field of each node - the extent to which it can gather information from its neighbors - is limited. As a result, the node representations may lack essential contextual information from the graph, particularly from nodes that are further away. This limitation prevents the GNN from leveraging the rich structural information available in the graph, leading to suboptimal performance in tasks requiring a broader understanding of the graph's topology [Barceló et al., 2020, Alon and Yahav, 2021].

Over-Squashing, Over-Smoothing and Under-Reaching similarly but in distinct ways highlight the delicate balance required in designing GNNs and specifically tuning its depth.

For a given task on a set of graphs, we call the *problem radius* r the distance which needs to be closed in order to be successful at that task. Using the message passing approach on a problem with radius $r = 3$, we would need at least 3 message passing layers in order to fit a model to the data [Alon and Yahav, 2021].

In this paper, we address the issue of Over-Squashing in scenarios with a large problem radius by altering the structure of Graph Neural Networks (GNNs). We modify the input graph's adjacency matrix in the final layer, enabling it to transmit previously gathered local information directly to distant nodes. This approach prioritizes the aggregation of local structures, a known strength of Message Passing Neural Networks (MPNNs), before disseminating this information over long distances in a single step. This strategy reduces the need for information to traverse through numerous intermediate nodes to reach its target, a process that often leads to Over-Squashing.

2 Related work

The main idea of this paper is an enhancement of the procedure proposed by Alon and Yahav [2021] in their paper "On the Bottleneck of Graph Neural Networks and its Practical Implications", where they describe the bottleneck of Over-Squashing and experiment with editing the adjacency in the last layer to contain all possible edges (fully adjacent).

With this, they achieve improvements of 42% on the QM9 dataset [Ramakrishnan et al., 2014, Ruddigkeit et al., 2012]. Further, they were able to consistently find improved accuracies for the biological benchmarks NCI1 [Wale et al., 2008] and ENZYMES [Borgwardt et al., 2005], as well as the VarMisuse dataset on computer programs [Allamanis et al., 2017].

Further, they trained the same model, but with a random subset of edges from the fully adjacent matrix. For 75%, 50% and 25% randomly selected edges they still achieved improvements over the regular approach (decreasing with the amount of edges) [Alon and Yahav, 2021].

More sophisticated techniques of selecting edges for the last layer were not further explored.

3 Methodology

3.1 Tree-NeighborsMatch

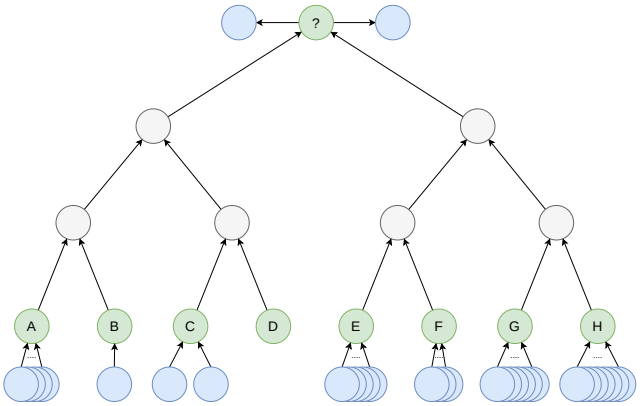

Figure 1: An example from the Tree-NeighborsMatch dataset with $depth = 3$ provided by Alon and Yahav [2021]. This figure features a green target node, marked with a question mark (?), positioned at the root of the tree. We also see eight green nodes labeled alphabetically. The green nodes are each connected to blue neighboring nodes. Every one of the labeled nodes has a distinct number of blue nodes, with only one matching the number of blue nodes with the target.

The Tree-NeighborsMatch dataset is crafted to provide insights into over-squashing effects within GNNs. It encapsulates a controlled environment where the graph's depth (problem radius r) directly influences the intensity of over-squashing. Through the lens of this dataset, we scrutinize the training accuracy of models to ascertain whether over-squashing impedes the fitting of long-range dependencies.

In the Tree-NeighborsMatch problem, a target node is situated at the root of a binary tree of depth K. The dataset simulates an exponentially-growing receptive field, with each leaf node represented by a one-hot vector labeling (A, B, C, etc.) and the number of its blue neighbors. The challenge is to predict the label for the target node based on the count of its blue neighbors, mirroring the label of the leaf node with an identical count (see figure 1).

Fitting a model to the Tree-NeighborsMatch problem, we try to maximize the accuracy on the training set. A high accuracy on the training set means, that the MPNN is able to fit the data. This is proof, that the representational power of the intermediate nodes is sufficient to aggregate the amount of data that is growing exponentially with the depth (i.e. Over-Squashing is not a problem).

Alon and Yahav [2021] tried several different MPNNs with depths between 2 and 8. For all different MPNNs there was the same trend, that after depth 3 or 4, the training accuracy dropped rapidly towards zero, indicating, that Over-Squashing poses a severe problem.

To differentiate the impact the long-range has from the effect of Over-Squashing, the authors have constructed a variation of the experiment for the depths of 4 to 8, where they created a tree of depth two, with each leaf being connected with a chain of intermediate nodes keeping the distance to the target the same (see figure 2). For these graphs, the diminishing performance was not observed, meaning, that the long-range alone is not the problem.

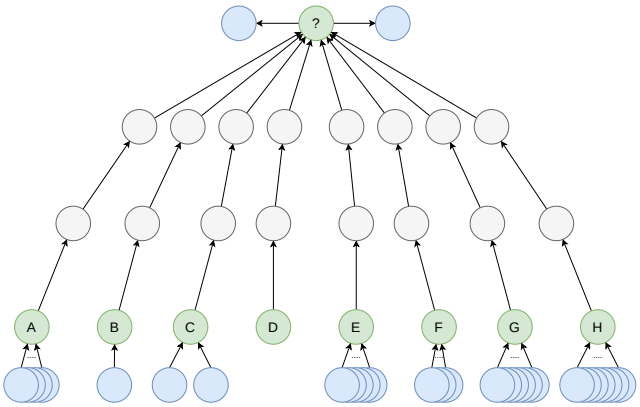

Figure 2: Graph to prove the isolated effect of Over-Squashing for the Tree-NeighborsMatch task. For this task training accuracies always reached 100%

3.2 Experiment

All the code for our experiments are provided on GitHub at `https://github.com/max-seeli/selective-long-range-connection-gnn.git`. They are based on the approach of Alon and Yahav [2021] which they implemented at `https://github.com/tech-srl/bottleneck/`.

We conducted an experiment with the same setup that Alon and Yahav [2021] chose for their analysis of the Tree-NeighborsMatch task. We only focused on the GCN [Kipf and Welling, 2016] and the GIN [Xu et al., 2018] with depth of 4, because our computational resources are limited, and the other GNN architectures (GGNN [Li et al., 2015], GAT [Veličković et al., 2017]) used for the experiment by Alon and Yahav [2021] already performed with a 100% accuracy at depth 4 and continued to show similar trends than the chosen architectures.

In our approach, we applied k-hop edge transformation for undirected graphs on the input graph. For an input graph $G = (V, E)$ where V is the set of vertices and E is the set of edges and for a given integer $k \geq 1$, the k-hop edge transformation of G is defined as $T_k(G)$ and results in a new graph $G' = (V, E')$, which retains the vertices V and where E' includes every edge (u, v) such that there exists a path of at least k hops from the vertex u to the vertex v in the original graph G.

Depending on the number of k, G' might have a substantially larger number of edges, as $T_k(G)$ degenerates into the complete graph $C(|V|)$ where $|V|$ is the number of nodes of the original graph:

$$\lim_{k \to 1} T_k(G) \to C(|V|) \tag{4}$$

With $N_k(u)$ representing the k-hop neighborhood (i.e. all nodes reachable within k-hops) of the vertex u, we can formally define T_k as:

$$T_k(G) = \bigcup_{u \in V} \{(u, v) | v \notin N_{k-1}(u)\} \tag{5}$$

Building upon the insights from the Tree-NeighborsMatch experiment and the concept of k-hop edge transformation, our approach introduces selective long-range edge inclusion in the final layer of the GNN. This method aims to mitigate the limitations of Over-Squashing in handling large problem radii by strategically enhancing the graph's connectivity in the later stages of message passing.

Our algorithm operates by dynamically altering the graph structure in the final layer of the GNN. The key idea is to initially focus on local structure learning through standard GNN layers and then, in the last layer, incorporate additional edges that enable direct long-range communication (see figure 3). This selective enhancement of connectivity is guided by a criterion that prioritizes edges which require information flow to nodes that are distant in the original graph structure.

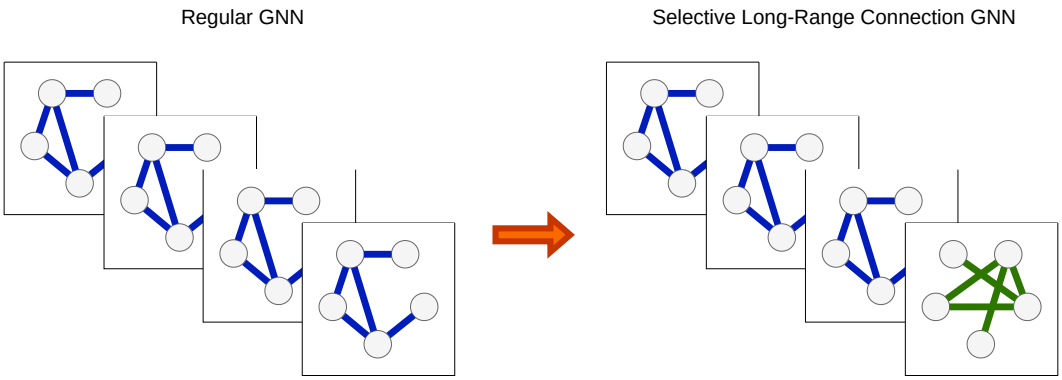

Figure 3: Visualization of the difference between the regular MPNN approach and our proposed process.

In our experimental setup, we employed a combination of specific hardware and software configurations to effectively conduct our compute intensive research.

For hardware, we used an Intel i9 CPU with 64 GB of RAM, primarily processing the graph transformations, which were all executed on the CPU. Additionally, an NVIDIA RTX 3080 graphics card was employed in combination with cuda drivers for the training of the GNNs parameters.

On the software side, our primary programming language was Python, chosen for its extensive range of libraries and tools widely used in machine learning and data processing. PyTorch served as our main machine learning library. For handling graph-structured data, we integrated PyTorch Geometric (PyG), an extension of PyTorch optimized for graph data. This library was instrumental in managing the graph structures and implementing the GNN layers required for our experiments. Additionally, NetworkX was used for performing k-hop edge transformations. The graph structures transformed by NetworkX were then converted and integrated into the PyTorch Geometric framework for further processing in GNNs.

For our comparative analysis with the findings of Alon and Yahav [2021], we adopted identical hyperparameters, which they had optimized across various elements such as different activation functions, layer normalization, batch normalization, residual connections, and batch sizes. We utilized the Adam optimizer, setting the learning rate at 10^{-3} and incorporating a decay factor of 0.5 upon reaching a plateau, which we defined as no improvement after 1000 epochs. The training duration was capped at $50,000$ epochs to maintain consistency in our methodology.

4 Results

For both the GCN and GIN we were able to achieve great improvements compared to the described results by Alon and Yahav [2021] (see table 1). The ability for the models to learn shows, that Over-Squashing can only play a minor role, as the information is being transported in a fashion that compresses it little enough that enough of the original information is received by the target node in order to make a correct classification.

Our results demonstrate significant enhancements in both the Graph Convolutional Network (GCN) and Graph Isomorphism Network (GIN) models, surpassing the findings reported by Alon and Yahav [2021], as detailed in Table 1. This advancement underscores the models' proficiency in learning and transmitting information effectively, with the issue of Over-Squashing appearing to have little impact, as the models are capable to condense information without significant loss and passing it to the target node which receives adequate data for accurate classification.

The learning process for both models resulted in both of them hitting the set limit of $50,000$ epochs, however, we can also see in figure 4, that they are close to convergence when hitting the epoch cap. Notably, they achieved accuracies of 99%, indicating a strong likelihood of reaching 100% accuracy if allowed a greater number of epochs.

Model	Baseline (Regular adjacency)	K-Hop adjacency
GCN	0.70	**0.9921**
GIN	0.77	**0.9991**
Relative:		+35%

Table 1: Accuracies for every tested model, comparing the approaches with the regular adjacency matrix for every layer and the k-hop edge transformation that is performed on the input graph before the last layer.

Interestingly, we observe a trend of overfitting in the Graph Convolutional Network (GCN) after $25,800$ epochs, as indicated by the declining test scores depicted in figure 4. In contrast, the Graph Isomorphism Network (GIN) does not exhibit this issue, although it demonstrates significantly higher variance in test scores throughout training. Given that our experiment prioritized data fitting (achieving high training scores) over generalization, these observations are not particularly concerning. Furthermore, we did not implement any overfitting prevention measures in this study.

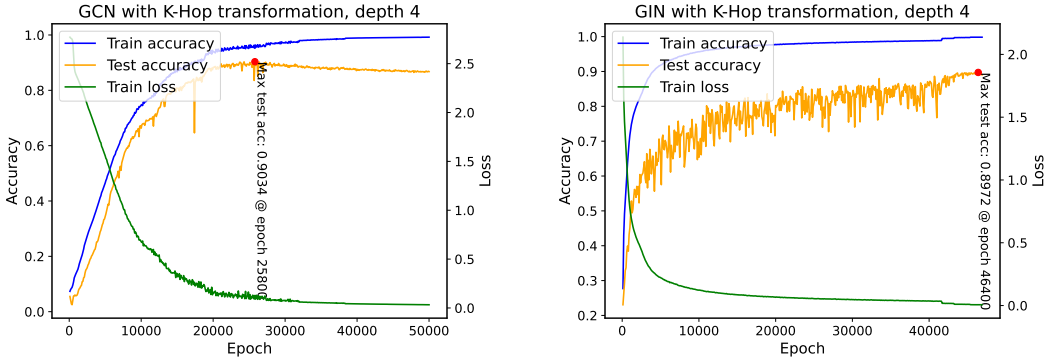

Figure 4: Learning process of the two different models

5 Discussion

The results from our experiment suggest that the selective long-range connection approach in Graph Neural Networks (GNNs) offers promising advancements in mitigating the issue of Over-Squashing. This is particularly evident from the significant improvements in accuracy observed in both the Graph Convolutional Network (GCN) and Graph Isomorphism Network (GIN) models. The high accuracy levels achieved by these models under our experimental conditions underscore the potential of this approach in handling complex graph-structured data.

It is important to note, however, that while the results are encouraging, the experiment could have been more comprehensive. Testing across a broader range of depths, rather than focusing solely on depth 4, would have provided a more robust understanding of the impact of selective long-range connections at varying levels of complexity and graph sizes. Additionally, experimenting with other edge selection methods could reveal alternative strategies that might be more effective or efficient in certain scenarios.

Another limitation of the study is the reliance on synthetic datasets, specifically the Tree-NeighborsMatch dataset. While this dataset effectively demonstrates the potential of our approach in a controlled environment, real-world datasets often present additional challenges and complexities not captured in synthetic scenarios. Testing our method on non-synthetic, real-world datasets would be a critical step in validating its practical applicability and effectiveness in real-world applications.

6 Conclusion

The research on selective long-range connections in Graph Neural Networks (GNNs) introduces a transformative approach to tackling the challenge of Over-Squashing. This method, which strategically alters the graph's structure in the final layer of the GNN, has demonstrated a substantial improvement in the learning capabilities of models, especially in scenarios involving large problem radii. This advance is significant as it addresses a key limitation in traditional GNN architectures.

The experimental results, obtained using the Tree-NeighborsMatch dataset and employing Graph Convolutional Network (GCN) and Graph Isomorphism Network (GIN) models, demonstrated significant improvements in model performance. Notably, both models showed a remarkable increase in accuracy, overcoming the limitations imposed by over-squashing and underscoring the effectiveness of our approach in enhancing global information flow while maintaining local feature distinctiveness. The results suggest that this method could lead to breakthroughs in the analysis and interpretation of extensive graph-structured data.

To delve deeper into the intricacies of graph neural networks and address some of the unresolved questions, it's important to consider several key areas for future research. A crucial aspect is the method used for selecting edges in the network, which appears to be central to the success of these models. One promising approach could involve methods that account for the local density of nodes or their degree (i.e., the number of connections each node has). These methods could then assign weights to these factors based on how likely it is that the information contained in these nodes would be impacted by Over-Squashing.

Another promising direction for research involves identifying the centers of dense clusters within a graph and strategically selecting edges that connect these dense clusters. The rationale behind this approach is that it would enable the model to capture and transmit local information from each cluster more effectively. Since the information from a dense cluster would be passed through the network only once (after the local structure is aggregated), this method could potentially enhance the efficiency and accuracy of the information flow within the graph neural network.

By exploring these methods and focusing on how to effectively select edges and nodes within a graph, future research could make new strides in improving the performance of graph neural networks, particularly in complex tasks involving large problem radii. This would not only address the current limitations but also expand the potential applications of these networks in various fields.

Overall, this research represents a significant step forward in the performance of GNNs, especially for extensive graph structures, and opens new avenues for exploration in the field.

References

Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. Learning to represent programs with graphs. *arXiv preprint arXiv:1711.00740*, 2017.

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=i80OPhOCVH2.

Pablo Barceló, Egor V Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, and Juan-Pablo Silva. The logical expressiveness of graph neural networks. In *8th International Conference on Learning Representations (ICLR 2020)*, 2020.

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

Qiaoyu Tan, Ninghao Liu, and Xia Hu. Deep representation learning for social network analysis. *Frontiers in big Data*, 2:2, 2019.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14:347–375, 2008.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.