Long-Range Dependencies & Bottlenecks in Graph Learning

Abhay Singh
Cornell University
as2626@cornell.edu

Sijia Linda Huang
Cornell University
sh837@cornell.edu

Franklin Tongxiang Deng
Cornell University
td268@cornell.edu

Abstract

Graph neural networks (GNNs) have shown to be effective in representation learning of structured data containing entities and their relations. To obtain long-range information, GNNs are typically stacked with multiple layers, which results in a “bottleneck" at the target nodes and the over-squashing of exponentially-growing information into fixed-size vectors. We experiment with various methods that attempt to capture long-range dependencies and study the impact of over-squashing on their performance. Specifically, we study the failure-case of methods performing prediction tasks depending on long-range information, wherein the graph fails to propagate messages from distant nodes. We consider a variety of methods to attempt to resolve this bottleneck: reducing the diameter of the graph by randomly rewiring edges in the graph, ignoring graph topology altogether by making the last layer fully adjacent, applying self-attention, propagating messages at multiple lengths, and the use of skip connections.

1 Introduction

Graph neural networks (GNNs) (Scarselli et al., 2009) have shown increasing popularity in recent years (Battaglia et al., 2018; Hamilton et al., 2017; Kipf & Welling, 2017; Xu et al., 2019) for representation learning tasks involving graph structure, particularly since many domains can be represented as graphs.

Each GNN layer is equivalent to a message-passing step (Gilmer et al., 2017), in which each node updates it own representation by computing a message by aggregating a representation of its neighbors. This message-passing mechanism of repeatedly propagating information between neighbors was shown (Alon & Yahav, 2020) to create a numerical information bottleneck when computing neighborhood aggregation. Typically, to model long-range interactions between nodes, GNNs are stacked with multiple layers to increase the receptive field of each node. However, this exponentially increases the number of nodes in the receptive field, and leads to the problem of *over-squashing*: information from this exponentially-growing receptive field is compressed into a fixed-length vector.

As a consequence, GNNs depending on long-range interactions in graph learning tasks suffer in performance and further generalize poorly as the model is only able to learn short-range signals present in training data. This work aims to compare, evaluate, and benchmark methods that aim to model long-range dependencies while avoiding the problem of over-squashing.

Preprint. Under review.

2 Background and Related Work

2.1 Graph Neural Networks

We first summarize common Graph Neural Network models (GNNs) and relevant notation, following Xu et al. (2019). Let $G = (V, E)$ denote a graph with input node feature vectors X_v for $v \in V$. There are two tasks we consider

1. *Node classification*, where each node $v \in V$ has a label y_v, and the aim is to learn a hidden representation h_v of each node v, such that the label of v can be predicted as $y_v = f(h_v)$
2. *Graph classification*, where given a set of graphs $\{G_1, \dots G_N\} \subseteq \mathcal{G}$ and their labels $\{y_1, \dots, y_N\} \subseteq \mathcal{Y}$, the aim is to learn a hidden representation h_G of each graph G, which can be used to predict its associated label, $y_G = g(h_G)$.

GNNs use the given graph structure $G = (V, E)$ and node features X_v to learn a representation of each node h_v, or a representation of the entire graph h_G. Many modern GNNs follow an iterative neighborhood aggregation strategy, where the representation of each node is updated by aggregating the representations of its neighbors. After k iterations of aggregation, the representation of a node captures structural information of its k-hop neighborhood within the graph. Formally, the k-th layer of a GNN is given by an aggregation step over neighboring nodes' features,

$$a_v^{(k)} = \text{AGGREGATE}^{(k)} \left(\{ h_u^{(k-1)} : u \in \mathcal{N}(v) \} \right)$$

where $\mathcal{N}(v)$ is the set of nodes adjacent to v, and then a combination step with the representation $h_v^{(k-1)}$ produced by the $(k-1)$-th layer,

$$h_v^k = \text{COMBINE}^{(k)} \left(h_v^{(k-1)}, a_v^{(k)} \right),$$

and AGGREGATE$^{(k)}$ and COMBINE$^{(k)}$ are parameterized neural networks. The feature vector of node v at the k-th iteration/layer is $h_v^{(k)}$, and is initialized with its input feature $h_v^{(0)} = X_v$. For node classification tasks, the representation of the final layer $h_v^{(K)}$ is used for prediction of each node. For graph classification, the function READOUT aggregates all node features of the final layer to obtain a representation for the entire graph h_G,

$$h_G = \text{READOUT} \left(\{ h_v^{(K)} : v \in G \} \right).$$

2.2 Bottleneck of GNNs

Our work is in part motivated by the over-squashing phenomenon that occurs in graph neural networks. At every layer of a graph neural network, each node computes a 'message' and sends it to its neighbors, updating its representation based on its received message and its previous representation. Alon & Yahav (2020) show that GNNs perform poorly for long-range tasks, which require multiple message passing steps (layers). They propose a NEIGHBORS-MATCH task: in a tree structure, predict the target node's label based by matching its number of neighboring nodes and those of the leaves. From the perspective of the target node, the rest of the graph may look like a tree, where the target node itself is the root. The number of layers required for messages to reach the target node equals the depth of this tree. However, the receptive field of the target node grows exponentially with the number of layers: for a tree of depth n, $O(degree^n)$ messages are needed to squash into the target node. This leads to a so-called 'bottleneck' at the target node and a phenomenon known as over-squashing: an exponential amount of information is squashed into a fixed-size vector.

To break this bottleneck, the paper proposed a modification to ignore graph topology in the last layer of the graph neural network, instead adding an edge between every pair nodes. Simply by doing so, performance of state-of-the-art models that suffer from over-squashing can be improved.

2.3 Implicit layers, DEQ and IGNN

In deep learning, models are usually made up of multiple layers, which are trained via backpropagation. A layer is then viewed as a differentible parametric function. Traditionally deep learning models use explicit layers based on explicit functions, which specifies how to compute the layer's output from the input. In contrast, $implicit$ layers specify the desired conditions for the layer's output to satisfy. Implicit layers include several advantages, including powerful representations, memory efficiency, simplicity as well as abstraction.

A class of emerging implicit layer models is the Deep Equilibrium (DEQ) Model (Bai et al., 2019). Their empirical evidence shows that hidden layers of many existing deep sequence models converge towards some fixed point, while DEQs are able to solve these fixed points directly. The iteration step of a DEQ can be expressed as

$$z^* = f(z^*, x, \theta)$$

where f is a generalized cell (e.g. LSTM), z^* is the fixed point, x is the input and θ represents the parameters of layers. In this way, the DEQ not only achieves the same effect as running an infinite depth network, but is also able to analytically backpropagate through the equilibrium point using implicit differentiation.

Following the idea of equilibrium models, long-term dependencies for GNNs can be captured more effectively by iterating the information passing step for an infinite number of times until convergence. Gu et al. (2020) proposes the Implicit Graph Neural Network (IGNN) not only to capture long-term dependencies by solving a fixed-point equilibrium equation, but also to resolve limitations in evaluation and training for recurrent GNNs. Specifically, they use Perron-Frobenius theory to deduce the sufficient condition for convergence, as well as leveraging implicit differentiation for training.

2.4 Higher-Order Neighborhoods

One of our methods is in part inspired by the MixHop (Abu-El-Haija et al., 2019) and SIGN (Rossi et al., 2020) models, which both aim to learn information from higher-order neighborhoods. Both schemes can be seen through the MixHop formulation

$$H^{(k+1)} = \Big\|_{j \in P} \sigma\left(\widehat{A}^j H^{(k)} W_j^{(k)}\right)$$

where $H^{(0)} = X$ initially, and $\widehat{A}^j$ is the jth power of the normalized adjacency matrix $\widehat{A} = D^{-1/2} A D^{1/2}$. In SIGN, we only calculate $H^{(k+1)}$.

The motivation of such a formulation is that it enables higher-order message passing, where nodes receive latent representations from their first-degree neighbors and from further N-degree neighbors at every message passing step. So, the model can learn to mix latent information from neighbors at various distances. In particular, this model can learn a two-hop delta operator

$$f\left(\sigma(\widehat{A}X) - \sigma(\widehat{A}^2 X)\right),$$

which common GNNs such as the Graph Convolutional Network (Kipf & Welling, 2017) cannot.

This is of interest as the incorporation of different powers of the normalized adjacency matrices in a GNN theoretically allows for message transmission between higher-order neighborhoods, thus providing a possible way to circumvent the GNN bottleneck. Some conducted experiments we conduct are motivated by this.

2.5 Virtual Nodes and Edges

Gilmer et al. (2017) also proposes modifying the graph topology directly, by adding a virtual master node and virtual edges. In particular, they add a separate "virtual" edge type for all pairs of nodes that are not connected. This can be implemented as a data preprocessing step and allows information to travel long distances during the propagation phase. They also experiment with a "master" node, which is connected to every input node in the graph with a special edge type. Fey et al. (2020) proposes a generalization of this. These methods can be seen as a way to reduce the length of the shortest path between any two nodes, and thus reduces the information bottleneck present in propagation between two previously distant nodes.

2.6 Self-Attention

The self-attention mechanism (Vaswani et al., 2017) has led to state-of-the-art results in many domains, including graph representation learning. Graph Attention Network (GAT) (Veličković et al.,

2018) and other models (Anonymous, 2021a,b) use attention mechanisms for GNNs, computing attention scores between pairs of nodes connected by an edge. Further, some works ignore graph topology altogether (Zhang et al., 2020) and compute attention scores between all nodes. In such a model, each node can 'attend' to the representations of its neighbors when computing a message, according to their attention scores.

Such models can be viewed to alleviate the over-squashing problem in two ways. The former models, by attending to certain neighbors during aggregation computation, a node's representation can ignore the irrelevant edges (by giving them very small attention weight). The latter by computing attention scores and directly modelling interactions between every node in the graph.

3 Methodology

3.1 Rewiring Edges

As mentioned in Gilmer et al. (2017), we add virtual edges as a preprocessing step. This reduces the expected diameter of the graphs, which facilitates information to travel long distances during the propagation phase. We explore various ways of doing so. One such way is to add edges randomly between nodes. Another way is to add edges between 'hubs', which are nodes with highest degree. This is similar to the notion of preferential attachment in link prediction tasks. These virtual edges are augmented with a fake edge feature, to represent that they were not originally present.

3.2 Fully Adjacent Layers

To tackle the bottleneck problem, we make the last layer fully adjacent (i.e. every node has an edge to every other node). Specifically, we construct two new matrices for every graph from the dataset. The first matrix is the fully adjacency index matrix, so that the (source node, root node) pair would include every possible combination of distinct nodes. The second matrix is the fully adjacency feature matrix. We expand the number of embedding dimensions for all involved features by 1, and denote the feature representations of the added "fake edges" to that extra dimension. Consequently, when the last layer of a graph is processed in the forward function, the fully adjacency index and feature matrices will be passed through instead of the original index and feature matrices. This method prevents over-squashing and eliminates the effect of the previously-existed bottleneck.

3.3 Transformer

Transformers (Vaswani et al., 2017) are a family of encoder-decoder neural network architectures that operate on sets of units, modelling long-range interactions between these units using self-attention. In recent years, transformers have achieved state-of-the-art results in many tasks, particularly in language modelling (Brown et al., 2020), and continue to be explored in other fields such as computer vision (Dosovitskiy et al., 2020).

In this work, we attempt to extend the transformer for the graph classification setting. As input, we treat the graph as fully-connected and input into it a set of nodes. The self-attention can then be thought of as a way of recovering edges in the graph. As a positional embedding for the nodes, we experimented with a regularized spectral embedding (Chaudhuri et al., 2012; Zhang & Rohe, 2018) coming from the leading k eigenvectors of the matrix $D_\tau^{-1/2}(A + \frac{\tau}{n}\mathbf{1}\mathbf{1}^T)D_\tau^{-1/2}$, where $\mathbf{1}$ is a vector of all ones, τ is a regularization parameter set to the average degree. To incorporate edge features simply in the model, we use a graph convolution on the output of the transformer, and then finally use global max-pooling across all node representations.

3.4 Different Powers of Adjacency Matrices in GCN

We incorporate adjacency matrices of various powers into each layer of a GNN, as per MixHop's proposal, in a bid to capture information from higher-order neighborhoods in each message-propagation step.

We begin with a vanilla GNN. In the forward pass for the kth GCN convolutional layer, we replace the one-hop self-propagation step with a k-hop self-propagation step. This effectively redefines the

GCN convolutional layer to the following:

$$H(i+1) = \sigma(\hat{A}^k H(0) W(i))$$

The motivation behind this amendment is to capture information from the kth order neighborhood in the kth convolution. Subsequently, these pieces of information can be combined by way of an attention mechanism, concatenation, or jumping knowledge.

3.5 Residual Connections and Intermediate Representations

A combination framework that may prove promising is jumping knowledge, first proposed by Xu et al. (2018).

For an arbitrary node v, the node embedding generated by each GCN convolutional layer yields intermediate representations, $h_v^{(1)}, ..., h_v^{(k)}$. Jumping Knowledge Network (JK-Net) enables each node to choose intermediate representations over which to perform the final combination step, thereby allowing the network to learn the neighborhood size that influences the node representation of each individual node.

We examine the JK-net combination technique and evaluate whether it proves effective for combating the bottleneck problem and enabling accurate predictions in graphs with long-range dependencies. Intuitively, this framework allows the model to determine an optimized neighborhood size, from which to gather information to inform the representation of each node, and effectively transmits information from those neighborhoods to said node.

4 Experiments

4.1 Open Graph Benchmark Datasets

The task of graph property prediction involves performing predictions such as classifications at the graph level. We focus on two OGB datasets intended for the graph prediction task, namely *ogbg-molhiv* and *ogbg-code* (Hu et al., 2020).

ogbg-molhiv is a small dataset, comprising graphs that represent molecules. Each node is an atom, and chemical bonds are represented by edges. With molecular properties identified as binary labels, the task is to predict the molecular properties of each target graph. Long-range dependencies feature heavily in this prediction task, as a molecule's properties may result due to atoms on opposite sides (Ramakrishnan et al., 2014).

A medium dataset, *ogbg-code* comprises graphs that represent Abstract Syntax Trees that represent method definitions in Python. Given input data for each graph comprising the method body as it is represented by AST and node features, graph property prediction aims to predict the tokens that constitute the method name.

4.2 Results of Making the Last Layer Fully Adjacent

We used the *ogbg-molhiv* dataset for this task.

Model Description	Best Validation Score	Test Score
Vanilla GCN	81.33	75.10
GCN + FA	84.64	77.46
Vanilla GIN	80.01	74.33
GIN + FA	82.69	77.32

Table 1: Performance of Fully Adjacent Last Layer on the *ogbg-molhiv* dataset

The Vanilla GCN and GIN achieves test scores of 75.10% and 74.33% respectively. In comparison, applying fully adjacent last layer to GCN and GIN yields test scores of 77.46% and 77.32%, which shows a clear improvement.

4.3 Results of Applying Different Powers of Adjacency Matrices

We incorporate various powers of adjacency matrices into GCNs, and run the graph property prediction task on *ogbg-molhiv*.

Model Description	Best Validation Score	Test Score
Vanilla GCN	81.22	75.48
Multihop GCN + JK	81.27	77.26
Multihop GIN	79.57	74.64
Multihop GIN + JK	80.33	75.36

Table 2: Multihop performance using GCN and GIN convolutional layers on *ogbg-molhiv* dataset, with and without using jumping knowledge in the combination step.

The vanilla GCN achieves a test score of 75.48% on the data. On this baseline, the incorporation of various powers of adjacency matrices into each GCN convolutional layer, which we term multihop, improves test score accuracy, attaining 77.26%. Multihop GIN, on the other hand, does not drastically enhance prediction performance.

4.4 Results of Transformer Experiments

We experiment with 3 variants of the Transformer model. In one, we take the output of the Transformer and use it as input of a GNN. Second, we do the reverse. And third, we incorporate a graph convolution in each layer of the Transformer. All results are on the *ogbg-molhiv* dataset.

Model Description	Test Score
Transformer$\rightarrow$GNN	0.72
GNN$\rightarrow$Transformer	0.60
Transformer with GraphConv	0.55

Table 3: Performance of Transformer experiments on *ogbg-molhiv* dataset

5 Discussion & Future Work

In this work, we have the initial explorations of various methods that aim to model long-range dependencies while avoiding the problem of over-squashing. We aim to submit a complete version of the work to a workshop (WWW 2021 `https://graph-learning-benchmarks.github.io/`, Feb 15), with further chosen datasets that have critical long-range dependencies and complete benchmarks of the discussed models. We hope to extend the work to propose better methods that can model long-range dependencies.

References

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Harutyunyan, H., Alipourfard, N., Lerman, K., Steeg, G. V., and Galstyan, A. Mixhop: Higher-order graph convolution architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning (ICML)*, 2019.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. *ArXiv*, abs/2006.05205, 2020.

Anonymous. Global attention improves graph networks generalization. In *Submitted to International Conference on Learning Representations*, 2021a. URL `https://openreview.net/forum?id=H-BVtEaipej`. under review.

Anonymous. Multi-hop attention graph neural network. In *Submitted to International Conference on Learning Representations*, 2021b. URL `https://openreview.net/forum?id=muppfCkU9H1`. under review.

Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. In *NeurIPS*, 2019.

Battaglia, P., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Çaglar Gülçehre, Song, H., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K. R., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. Relational inductive biases, deep learning, and graph networks. *ArXiv*, abs/1806.01261, 2018.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020.

Chaudhuri, K., Chung, F., and Tsiatas, A. Spectral clustering of graphs with general degrees in the extended planted partition model. In *Conference on Learning Theory*, pp. 35–1, 2012.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

Fey, M., Yuen, J.-G., and Weichert, F. Hierarchical inter-message passing for learning on molecular graphs. *ArXiv*, abs/2006.12179, 2020.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL `http://proceedings.mlr.press/v70/gilmer17a.html`.

Gu, F., Chang, H., Zhu, W., Sojoudi, S., and Ghaoui, L. Implicit graph neural networks. *ArXiv*, abs/2009.06211, 2020.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *ArXiv*, abs/2005.00687, 2020.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Ramakrishnan, R., Dral, P. O., Rupp, M., and von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

Rossi, E., Frasca, F., Chamberlain, B., Eynard, D., Bronstein, M., and Monti, F. SIGN: Scalable inception graph neural networks. *arXiv:2004.11198*, 2020.

Scarselli, F., Gori, M., Tsoi, A., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20:61–80, 2009.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *ICML*, 2018.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.

Zhang, J., Zhang, H., Xia, C., and Sun, L. Graph-bert: Only attention is needed for learning graph representations, 2020.

Zhang, Y. and Rohe, K. Understanding regularized spectral clustering via graph conductance. In *Advances in Neural Information Processing Systems*, pp. 10631–10640, 2018.