Expressive Power, Generalization, and Optimization of Graph Neural Networks: A Survey

Yusu Qian yq729@nyu.edu
Courant Institute of Mathematical Sciences
NYU

Muge Chen mc7645@nyu.edu
Courant Institute of Mathematical Sciences
NYU

Yibo Liu yl6769@nyu.edu
Courant Institute of Mathematical Sciences
NYU

Fei Xu jx1260@nyu.edu
Courant Institute of Mathematical Sciences
NYU

Abstract

Graph Neural Networks(GNNs) is widely used in solving graph-based problems and has shown promising resutls. Despite their empirical success, the are limited understanding on the characteristics and limitations of GNNs. The major issues in GNNs are their representational ability, generalization ability, and optimization techniques. Recently, many theories are proposed to address the limitations, and many models are proposed to overcome the limitations. In this survey, we provide a comprehensive review on the three aspects.

Keywords: Graph Neural Network, Expressive Power, Generalization Ability, Over-Smoothing

1. Introduction

Recently, analysis of graph data with machine learning methods is gaining more attention, because this non-Euclidean data structure can be used as denotation for many systems in lots of learning tasks, e.g. social network(Hamilton et al., 2017), chemo-informatics(Gilmer et al., 2017), knowledge base(Hamaguchi et al., 2017) and n-body problems(Battaglia et al., 2016). GNNs are inspired by convolutional neural networks(CNNs) and graph embedding. CNNs can effectively extract the localized spatial features and construct highly expressive representations. It is intuitively reasonable to adapt CNNs to graphs, but CNNs are originally designed for Euclidean data. Graph embedding learns the low-dimensional representations of the nodes and edges, but the number of its parameters is linear to the size of the graph and it lacks flexibility to adapt to a dynamic graph. Taking the advantages of

both, GNNs extract information from the graph structure using neighborhood aggregation scheme and learn representations of nodes. The existing GNNs models and their variants have shown success in node classification, link prediction, and downstream applications. Zhou et al. (2018) provided a comprehensive survey on GNNs.

Despite the empirical successes of GNNs, we still have limited understanding of GNNs' representational capacity, generalization ability, and effective optimization strategies.

As for representational ability, Xu et al. (2018a) first proposed a framework to analyze the expressive power of GNNs. They mathematically formalized with the framework the insight that a GNN can have as large discriminative power as the Weisfeiler-Lehman(WL) test if the GNN's aggregation scheme is highly expressive and can model injective functions. Their results showed that message passing GNN cannot distinguish between graphs that are indistinguishable by the 1-WL test, demonstrating GNNs' expressive ability is not so strong as the universal approximation ability of multi-layer perceptrons(Park and Sandberg, 1991). On the other hand, Sato et al. concluded that GNNs are at most as powerful as distributed local algorithms. Thus, many provably powerful GNNs are proposed(Maron et al., 2019a). Sato (2020) provided a comprehensive survey of the expressive power of GNNs with an inductive analyze of the relation between GNN and WL algorithms and distributed local algorithms.

Generalization ability of GNNs means how well a model can predict labels for unseen graphs, and it is measured by generalization error. Some studies tried to derive the bounds on generalization error (Belkin et al., 2004; Verma and Zhang, 2019). Models(Xu et al., 2018b) are proposed to improve the generalization ability.

In terms of GNNs' optimization, over-smoothing and over-fitting of GNNs are the major issues to solve. GNN models are shallow due to over-smoothing, restricting its ability to extract high-layered features on a graph. Li et al. (2018) first pointed out that over-smoothing in GNNs is caused by Laplacian smoothing. Consequently, a bunch of optimization methods emerged, aiming to make the GNNs deeper. Some heuristic models borrow ideas from CNN. The models used to train for deeper CNNs in computer vision, e.g. ResNet(He et al., 2016), DenseNet(Huang et al., 2017), Inception(Szegedy et al., 2014), are adapted to GCNs(Li et al., 2019; Szegedy et al., 2014). Other techniques include changing the normalization method(Chiang et al., 2019), adding regularization to the training objective(Chen et al., 2019a), and applying the dropout mechanism to edges(Rong et al., 2019). Besides the improvements in results, there are also theoretical analyses from a topological perspective to show why these techniques work(Chen et al., 2019a).

This survey is organized as the following: 2 presents the review of GNNs' representational power. First, we introduce the work describing a mathematically formalized framework for representational ability, then the studies from the two perspectives are introduced respectively: (1) the ability to distinguish non-isomorphic graphs, (2) the ability to approximate permutation-invariant functions. 3 provides a review of GNN's generalization ability. First we introduced the theory giving the generalization bounds. Then we describe the studies focusing on the generalization ability of different models and the improvement strategies. 4 discusses the optimization techniques for GNNs. First we show the work that gives a mathematical illustration of the over-smoothing phenomenon. Then various optimization methods from different intuitions are shown and their advantages and disadvantages are given. 5 states the remaining problem and the future work.

2. Representational Power of Graph Neural Network

Graph Neural Networks (GNNs) are effective for representation learning of graphs. Many GNN variants have been presented and have achieved SOTA results on tasks such as node and graph classification from many fields. Hence, an interesting problem has been proposed and investigated: understanding of GNNs representational power and limitations. Here, we investigated these works from different perspectives.

2.1 Expressive Power of GNNs: A Theoretical Framework Review

Xu et al. (2018a) first proposed a framework to study the representational power of GNNs. The framework is inspired by the close connection between GNNs and the Weisfeiler-Lehman (WL) graph isomorphism testWeisfeiler and Leman (1968). Figure 1 from Xu et al. (2018a) illustrates their idea.

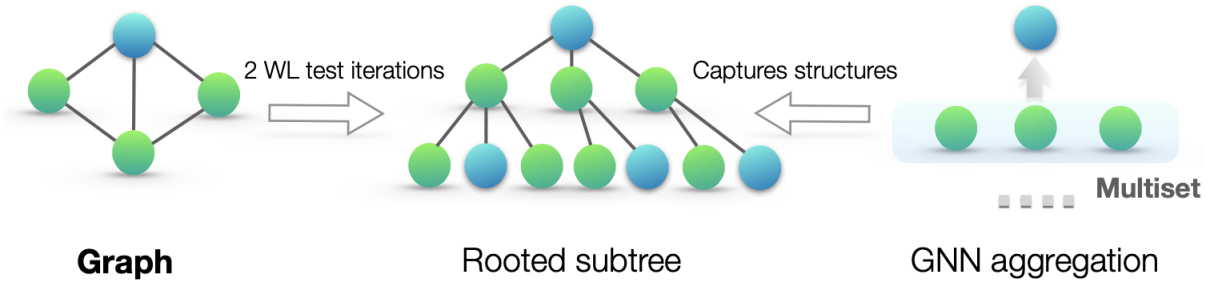

Figure 1: An overview of the framework. Middle panel: rooted sub-tree (at the blue node) which the WL test uses to discriminate different graphs. Right panel: The GNN can capture the rooted sub-trees in a recursive manner if the full multiset of neighborhoods was captured by a GNN's aggregation function, and achieve the power level of the WL test.

In a nutshell, they analyzed the expressive power of a GNN when it maps two nodes to the same position in the vector space. Then to the specific question whether a GNN maps two neighborhoods to the same embedding. And the answer is obviously: A maximally powerful GNN would never map two different neighborhoods to the same representation. Hence, its aggregation scheme must be injective under that assumption. Last, a GNN's aggregation scheme was abstracted as a class of function over multisets which their networks can represent, and then analyze whether they can represent injective multiset functions.

Following this work, many other frameworks have also been presented, which will be characterized in bellow.

2.2 Ability to Distinguish (Sub-)Graphs

One perspective of studying the representational capacity of GNNs via their ability to distinguish graphs which are non-isomorphic in these works. As mentioned in above, Xu et al. (2018a) designed a framework to investigate how powerful are GNNs, and show that

any aggregation-based GNN is at most as powerful as the WL graph isomorphism test (Weisfeiler and Leman (1968)) to distinguish different graphs.

Besides Xu et al. (2018a), Morris et al. (2019) also showed that GNNs have the same expressiveness as the 1-WL in distinguishing non-isomorphic graphs and sub-graphs also from a theoretical viewpoint. And proposed an architectures k-GNNs, which are neural architectures based on the k-dimensional WL algorithm (k-WL), which are strictly more powerful than GNNs. The models can capture structural information that is not visible at the node-level. Also, Chen et al. (2020) examined the representation power of GNNs in distinguishing matching-count and containment-count substructure counting, and they show both positive and negative results for GNNs. For instance, Message Passing Neural Networks (MPNNs), 2-Weisfeiler-Lehman (2-WL), and 2-Invariant Graph Networks (2-IGNs) cannot matching-count any structures with 3 or more nodes but can containment-count of star-shaped structures. While k-IGNs and k-WL can containment-count and matching-count.

2.3 Ability to Approximate Functions

There are works studying the expressive power of GNNs from another viewpoint, these works do so by studying the ability of GNNs to approximate permutation-invariant functions on graphs. Maron et al. (2019b) paid attention to the fundamental question: Can GNNs approximate any(continuous) invariant function. And they generalize the case where $G \leq S_n$ (an arbitrary subgroup of the symmetric group) that acts on $\mathbb{R}^n$ by permuting coordinates. They show that universal G-invariant networks can be obtained if high-order tensors are allowed inside the model. And for some groups G to obtain universality, the networks must contain higher-order tensors. Further, a necessary condition has been proved for the universality of invariant GNNs. Following Maron et al. (2019b)'s result, Keriven and Peyré (2019) proved it by an alternative proof, which bases on the Stone-Weierstrass theorem and extends the work by considering the equivariant case. Additionally, they show that a function defined on graphs of a varying number of nodes can be uniformly well approximated by a GNN bound with a single set of parameters.

More interesting, Chen et al. (2019b) dedicated to understanding the expressive power of GNNs by connecting this perspective with the above-mentioned one. That is, they proved equivalence between the ability to distinguish non-isomorphic graphs and the capacity to approximate permutation-invariant functions on graphs. Besides, they proposed an architecture of the expressive power of GNNs with the language of sigma-algebra, which not only considered the first perspective but also considered the second one. Hence, the representation power of variant GNNs measured by this way can be compared with other methods on graphs. They show that order-2 Graph G-invariant networks fail to discriminate non-isomorphic graphs with the same degree. To address that issue, a new architecture called Ring-GNN has been designed. It should be noted that Chen et al. (2020)'s work for understanding the ability of GNNs also based on proven an equivalence between approximating graph functions and distinguishing graphs.

However, for practical GNNs, which are not universally approximating, it is worth for us to understand what they can and cannot do. As Loukas (2019) pointed out that GNNs are Turing universal under assumptions on their node attributes, width, depth, and layer

expressiveness, but they can lose power strikingly when their depth and width is limited. We believe more works were needed in terms of understanding the representation power of variant GNNs used in practice as guidance in searching for more powerful models.

In the above, we investigated the works measure the expressive power of GNNs. Their generalization ability will be reviewed below.

3. Generalization Ability of Graph Neural Network Models

3.1 Generalization Error

In machine learning, generalization error measures how well a model performs on previously unseen data. The generalization error of a function f which has x and y as its possible values is calculated as:

$$I[f] = \int_{X \times Y} V(f(x), y)\, \rho(x, y) dx dy \tag{1}$$

Zhang et al. (2016) showed that traditional methods could not reason why large neural network models generalize well. The generalization ability of GNNs has not been extensively studied. Ando and Zhang (2006) used geometric properties of the graph and Laplacian regularization to derive margin-based generalization bounds.

3.2 Generalization Bounds for GNNs

Belkin et al. (2004) used the notion of algorithmic stability to derive bounds on generalization error. This is related to the structural invariants of the graph. In the same line of work, Verma and Zhang (2019) derived generalization bounds for GCNN models by studying the stability of them, as the stability largely affects generalization. Another reason they used stability to derive generalization bounds is that traditional methods towards this goal are either difficult to implement or not applicable for neural networks. Garg et al. (2020) quantitatively studied the generalization ability of GNNs. They focused on binary classification. First, they analyzed how in a fixed tree, changes in shared variables of the root node's embedding bring about effects quantitatively. To achieve this, they used the maximum effect across each subtree to recursively bound the effect on them. They calculated the generalization bound of GNN models as follows. They considered locally permutation invariant GNNs. In each layer l, they aggregate the embeddings of the neighbors of node v of a given input graph using aggregation function $\rho : \mathbb{R}^r \to \mathbb{R}^r$ to update its embedding $h_v^\ell \in \mathbb{R}^r$. Among multiple possible types of update, they chose a mean field update.

$$h_v^\ell = \phi\left(W_1 x_v + W_2 \rho\left(\sum_{u \in N(v)} g\left(h_u^{\ell-1}\right)\right)\right) \tag{2}$$

Here we omit the further details of their approach and list their primary results. In the table below, m represents sample size, or the number of graphs in the set $\mathcal{G}$, r represents dimensions, L represents length, and d is a branching factor. The dependence of the generalization ability of GNN models is on these factors is similar in nature to that in RNN models. The authors further made comparisons with other bound methods. Compared with VC-bounds for GNNs, this bound is significantly tighter.

$\mathcal{C}$	GNN (Garg et al., 2020)	RNN (Chen et al. (2019a))
$< 1/d$	$\tilde{\mathcal{O}}\left(\frac{rd}{\sqrt{m\gamma}}\right)$	$\tilde{\mathcal{O}}\left(\frac{r}{\sqrt{m\gamma}}\right)$
$= 1/d$	$\tilde{\mathcal{O}}\left(\frac{rdL}{\sqrt{m\gamma}}\right)$	$\tilde{\mathcal{O}}\left(\frac{rL}{\sqrt{m\gamma}}\right)$
$> 1/d$	$\tilde{\mathcal{O}}\left(\frac{rd\sqrt{rL}}{\sqrt{m\gamma}}\right)$	$\tilde{\mathcal{O}}\left(\frac{r\sqrt{rL}}{\sqrt{m\gamma}}\right)$

3.3 Generalization Ability of Different Classes of GNNs

Wu et al. (2019) compared between early spectral-based models for graph data and spatial-based models in terms of generalization ability. Spectral-based models work well on data with the same structure but cannot be generalized to different structures. In comparison, spatial-based models can share parameters across different structures as graph convolution is performed on nodes locally. It is not only better at generalization but also more efficient to represent nodes in a graph using local neighborhood information instead of the entire graph and train locally.

3.4 Improve Generalization Ability of GNNs

Xu et al. (2018a) showed that increasing the number of iterations refines and globalizes node representations when training GNN models. If the goal is for the model to achieve discriminating power, then the model needs to be trained for a sufficient number of iterations. However, a drawback is that the generation ability of features decreases in the process. Thus, to achieve the best of both worlds, Xu et al. (2018a) proposed keeping information of features from all iterations during the training process to produce GNN models with strong expressive power and generalization ability. Their analysis did not directly speak about the generalization ability of their model and other classes of GNN models, but they argued that the generalization ability can be inferred from test accuracy of the models, and that it is reasonable to assume that models with strong expressive power are better at generalizing because graph structures are captured better. The model they proposed, Graph Isomorphism Network (GIN), meets the requirements in Theorem 1. The model, while having a simple architecture, generalizes the WL test.

Theorem 3 (Xu et al., 2018a). Let A : $\mathcal{G} \to \mathbb{R}^d$ d be a GNN. With a sufficient number of GNN layers, A maps any graphs G_1 and G_2 that the Weisfeiler-Lehman test of isomorphism decides as non-isomorphic, to different embeddings if the following conditions hold:

a) A aggregates and updates node features iteratively with

$$h_v^{(k)} = \phi\left(h_v^{(k-1)}, f\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right)\right) \tag{3}$$

where the functions f, which operates on multisets, and ϕ are injective.
b) A's graph-level readout, which operates on the multiset of node features $\left\{h_v^{(k)}\right\}$, is injective

4. Optimization for GCN

Though there are lots of variant of GNNs emerging nowadays, here we take the original vanilla Graph Convolution Networks(GCNs) as a base case.

Compared to CNNs, earlier GCNs suffer from over-fitting, over-smoothing, and vanishing gradient problems, thus their models are shallow (number of layers is usually smaller than 3). In recent years, several methods have been proposed to build deep GCNs.

Li et al. (2018) were the first to call attention to the over-smoothing problem. Having shown that the graph convolution is a type of *Laplacian smoothing*, they proved that after repeatedly applying *Laplacian smoothing* many times, the features of the nodes in the (connected) graph would converge to similar values—the issue coined as "over-smoothing". But unfortunately, they never proposed any method to address it.

4.1 Why GNNs Have Over-smoothing Problems

To understand why there is over-smoothing in GNNs, we have to understand how it works. The propagation rule of a simplified GNN model is:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}\Theta^{(l)}\right) \tag{4}$$

$$H^{0} = X \tag{5}$$

where $H^{(l)}$ is the matrix of activations in the l -th layer, and $H^{(0)} = X, \Theta^{(l)} \in R^{c\times f}$ is the trainable weight matrix in layer l, σ is the activation function, e.g., $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$ and $\tilde{A}$ is the adjacency matrix for the graph.

Li et al. (2018) proved that

$$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}$$

is a special form of Laplacian smoothing – symmetric Laplacian smoothing. The Laplacian smoothing computes the new features of a vertex as the weighted average of itself and its neighbors'. That explains why when GCNs go deeper, the states will tend to be the same - which we called "oversmoothing".

Graph convolutions essentially push representations of adjacent nodes mixed with each other, such that, if extremely we go with an infinite number of layers, all nodes' representations will converge to a stationary point, making them unrelated to the input features and leading to vanishing gradients.

As mentioned before, there are several methods to alleviate this issue.

4.2 CNN Inspired Methods

4.2.1 Residual related methods

In the early works, encouraged by the success of residual connections(He et al., 2016) solving the vanishing gradient problem in CNN used in computer vision, Kipf and Welling (2016) introduced residual connections between hidden layers to GCN.

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}\Theta^{(l)}\right) + H^{(l)} \tag{6}$$

The experiment shows that training without residual connection becomes difficult for models more than 7 layers, while the model with residual connections of 10 layers could still converge. Although residual connection solves the gradient vanishing problem in training, the performance of deeper models is not significantly improved. Even GCNs with residual connections, the models do not always achieve the same performance as the two-layer GCN, as Xu et al. (2018b) pointed out.

Inspired by DenseNet(Huang et al., 2017), a stronger model evolved from ResNet(He et al., 2016) in computer vision, Li et al. (2019) proposed DenseGCN to exploit dense connectivity among layers, which integrate information into the next layer from all layers instead of only the previous layer. Figure 2 shows the different types of connections between layers.

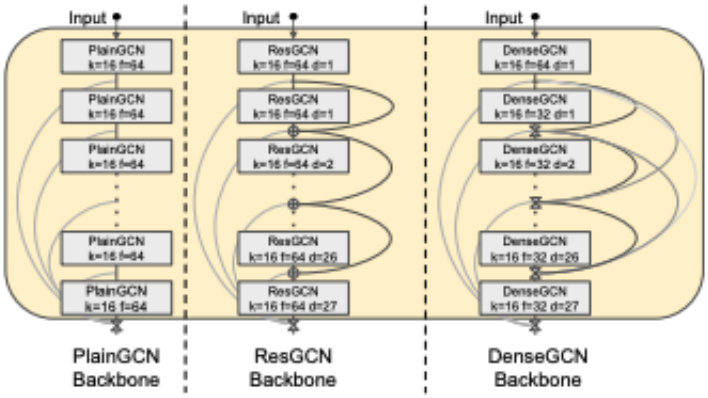

Figure 2: The framework of GCNs with different layer connections.(Li et al., 2019)

Particularly, the propagation rule of DenseGCN is defined as the following:

$$Z^{(l)} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}\Theta^{(l)} \tag{7}$$

$$H^{(l+1)} = \mathcal{T}(\sigma(Z^{(l)}), H^{(l)}) \tag{8}$$

$$= \mathcal{T}(\sigma(Z^{(l)}), ..., \sigma(Z^{(0)}), H^{(0)}) \tag{9}$$

Where the operator $\mathcal{T}$ is a vertex-wise concatenation function that densely fuses the input graph with all the intermediate GCN layer outputs. However, with the model goes deeper, pooling is more likely to cause information loss. Thus, they also introduced dilated convolutions to GCNs, which is originally used as an alternative of consecutive pooling layers on dense prediction tasks to alleviate spatial information loss caused by pooling operation. Their experiments showed that their model can be as deep as 56 layers with significant improvement on benchmarks. They also show that dilated graph convolutions help to gain a larger receptive field without loss of resolution. This work successfully incorporate the concepts in CNN to GCN, which encourages further studies into adapting other CNN architectures into GCN.

4.2.2 Wider instead of Deeper

Some studies do not insist on getting deeper on a single model, instead, they make it wider. Inspired by the idea of Inception(Szegedy et al., 2014) widely used in training CNN, which

stacks the pooling output of the size 1×1, 3×3, 5×5 together, Abu-El-Haija et al. (2018) proposed a multi-scale GCN, called N-GCN. It trains multiple instances of GCNs over node pairs of different distances in random walks, which can be understood as reception fields of different scales, and learns a combination of the instance outputs. To be particular, the propagation is given by:

$$N - GCN = softmax(\mathcal{T} \begin{pmatrix} GCN(A^{-0}, X; \theta^0) \\ GCN(A^{-1}, X; \theta^1) \\ ... \\ GCN(A^{-n}, X; \theta^n) \end{pmatrix} W) \tag{10}$$

Where $GCN(A, X; \theta) = \sigma(AX\theta)$, $\mathcal{T}(\cdot)$ is the concatenated feature.

This method avoids the over-smoothing problem while maintaining the model representational ability.

4.3 Improve Normalization

According to the analysis on 4.1, we can see that in the k^{th} layer propagation, the elements in the graph convolutional filter $\mathcal{F}(\mathcal{G}) = (\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}})^k$ with large k tends to converge to the same value. So a straightforward idea is to change the convolutional filter itself. Over-smoothing means the neighbours of different steps are considered evenly, but intuitively, the neighbors nearby should contribute more than distant nodes. Thus Chiang et al. (2019) proposed an alternative convolution kernel, which amplifies the diagonal parts of the adjacency matrix A to put more weights on the representation from previous layer, for example, to add an identity to A:

$$H^{(l+1)} = \sigma\left((\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} + I)H^{(l)}\Theta^{(l)}\right) \tag{11}$$

Further more, to take the numbers of neighbors into account, they proposed a modified version of normalization:

$$A' = (D + I)^{-1}(A + I) \tag{12}$$

$$H^{(l+1)} = \sigma\left((A' + \lambda diag(A'))H^{(l)}\Theta^{(l)}\right) \tag{13}$$

Their experiments show the 8-layer GCN convergences with a "diagonal enhancement" filter, while other models do not converge.

While no other work reporting the modified filter shows better performance on deep models, Wang et al. (2020) explored the performance of different filters and proposed a learnable filter. Thus for future work, we could also explore which kind of filter is good for deep GCN or what properties a filter should has.

4.4 From Dropout to DropEdge

DropEdge (Rong et al., 2019) is proposed as a general approach for all kinds of GNN models. It is similar to Dropout mechanism, and it is very simple:

The term "DropEdge" refers to randomly dropping out a certain rate of edges of the input graph for each training time.

First, it can be considered as a data augmentation technique. Which is useful to avoid over-fitting. Second, by removing edges from the input graph, the connections among nodes become more sparse, and hence avoiding over-smoothing to some extent when GCN goes very deep.

Rong et al. (2019) then proved that DropEdge will reduce oversmoothing by prove the following theorem:

Theorem 1 *We denote the original graph as $\mathcal{G}$ and the one after dropping certain edges out as $\mathcal{G}'$. Given a small value of ϵ, we assume $\mathcal{G}$ and $\mathcal{G}'$ will encounter the ϵ -smoothing issue with regard to subspaces $\mathcal{M}$ and $\mathcal{M}'$, respectively. Then, either of the following inequalities holds after sufficient -edges removed.*

- *The relaxed smoothing layer only increases:* $\hat{l}(\mathcal{M}, \epsilon) \leq \hat{l}\left(\mathcal{M}', \epsilon\right)$
- *The information loss is decreased:* $N - \dim(\mathcal{M}) > N - \dim\left(\mathcal{M}'\right)$

Hence we can state that DropEdge either reduces the convergence speed of over-smoothing or relieves the information loss caused by it.

Though it prevents over-fitting and alleviates over-smoothing, it may lack generalization for other applications because some information has been dropped.

4.5 Regularization and Dynamic Edge

Chen et al. (2019a) proposed 2 methods to alleviate the over-smoothing issues in GNNs : (1) MADReg which adds a MADGap-based regularizer to the training objective; (2) AdaEdge which optimizes the graph topology based on the model predictions.

They argue that one key factor leading to the over-smoothing issue is the over-mixing of information and noise. The interaction message from other nodes may be either helpful information or harmful noise. For example, in the node classification task, intra-class interaction can bring useful information, while inter-class interaction may lead to indistinguishable representations across classes. To measure the quality of the received message by the nodes, they defined the information-to-noise ratio as the proportion of intra-class node pairs in all node pairs that have interactions through the GNN model. And they think that low information-to-noise ratio will lead to over-smoothing.

They proposed that the low information-to-noise ratio is caused by the discrepancy between the graph topology and the objective of the downstream task. In the node classification task, if there are too many inter-class edges, the nodes will receive too much message from nodes of other classes after several propagation steps, which would result in over-smoothing. To prove their assumption, they optimize the graph topology by removing inter-class edges and adding intra-class edges based on the gold labels, which proves very effective in relieving over-smoothing and improving model performance.

The idea of removing edge here is similar to DropEdge. But DropEdge removes edge randomly but here they focus on removing inter-class edges and adding intra-class edges. This can be seen as a regularization to graph topology.

5. Limitation: What Graph Neural Networks Cannot Learn

There is still a lot to understand about graph neural networks, but there were quite a few important results about what GNNs cannot learn.

Loukas (2019) provided an interesting view of GNN from the direction of theoretical computer science. He studied the expressive power of message-passing graph neural networks. This paper stated that GNNs can compute any function on its input that is computable by a Turing machine under sufficient conditions. And it proved that when the product of depth and width $d \cdot w$ of the network is restricted, then GNNs will lose a significant portion of their power. For example, if we limit the product of depth and width to a constant number, then this model will not be able to detect whether a graph contains a cycle of a specific length. This paper even gives several problems that are deemed impossible unless the product of a GNN's depth and width exceeds a polynomial of the graph size; this dependence remains significant even for tasks that appear simple or when considering approximation, which shows the limitation of GNNs from an interesting point.

To understand the empirically observed phenomena that deep non-linear graph NNs do not perform well, Oono and Suzuki (2019) analyzed their asymptotic behaviors by interpreting them as a dynamical system. They proved that if the weights of a GCN satisfy conditions determined by the graph spectra, the output of the GCN carries no information other than the node degrees and connected components for discriminating nodes when the layer size goes to infinity. They showed that when the graph is sufficiently dense and large, many GCNs suffer from the information loss:

Theorem 2 *Consider GCN on the Erdős–Rényii graph $G_{N,p}$ such that $\frac{\log N}{Np} = o(1)$ as a function of N. For any $\epsilon > 0$, if the supremum s of the maximum singular values of weights in the GCN satisfies $s < \frac{1}{7}\sqrt{\frac{Np-p+1}{\log(4N/\epsilon)}}$, then, for sufficiently large N, the output of this GCN exponentially approaches a matrix that only has information about connected components and node degrees with probability at least $1-\epsilon$. Which means in this case the GCN can not distinguish the nodes with same node degrees and within the same connection components.*

However, all analyzes above are merely theoretical results based on uncommon conditions. For instance, Oono and Suzuki (2019) pointed out they can not extend their result for other activation functions such as the sigmoid function or Leaky ReLU. And the real world graphs are not dense.

6. Conclusion

In recent years, graph neural networks have been widely used as an effective and powerful tool for graph-related machine learning tasks. The significant progresses benefit from expressive power, model flexibility, and training strategies. In this survey, we conduct a review on the three above fundamental aspects in GNNs: representational ability, generalization ability, and optimization techniques. For each division, we first clarify the specific meaning of the issue especially in graph networks, next we describe the most fundamental work that gives the theoretical explanation to the issue, then we introduce further studies from various perspectives or initiations. Finally, we point out the remaining challenges for GNN theory and the possible focuses for future work.

References

Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee. N-gcn: Multi-scale graph convolution for semi-supervised node classification. 2018.

Rie Kubota Ando and Tong Zhang. Learning on graph with laplacian regularization. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, page 25–32, Cambridge, MA, USA, 2006. MIT Press.

Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *CoRR*, abs/1612.00222, 2016. URL `http://arxiv.org/abs/1612.00222`.

Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In *COLT*, 2004.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view, 2019a.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. In *Advances in Neural Information Processing Systems*, pages 15868–15876, 2019b.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *arXiv preprint arXiv:2002.04025*, 2020.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. 2019.

Vikas K. Garg, Stefanie Jegelka, and Tommi S. Jaakkola. Generalization and representational limits of graph neural networks. *ArXiv*, abs/2002.06157, 2020.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212, 2017. URL `http://arxiv.org/abs/1704.01212`.

Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. Knowledge transfer for out-of-knowledge-base entities : A graph neural network approach. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1802–1808, 2017. doi: 10.24963/ijcai.2017/250. URL `https://doi.org/10.24963/ijcai.2017/250`.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

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

Gao Huang, Zhuang Liu, Laurens Van De Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. 2017.

Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. In *Advances in Neural Information Processing Systems*, pages 7090–7099, 2019.

Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2016.

Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *CoRR*, abs/1801.07606, 2018. URL `http://arxiv.org/abs/1801.07606`.

Andreas Loukas. What graph neural networks cannot learn: depth vs width. *ArXiv*, abs/1907.03199, 2019.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. 2019a.

Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. *arXiv preprint arXiv:1901.09342*, 2019b.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification, 2019.

J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification, 2019.

Ryoma Sato. A Survey on The Expressive Power of Graph Neural Networks. *arXiv e-prints*, art. arXiv:2003.04078, March 2020.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Approximation ratios of graph neural networks for combinatorial problems.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL `http://arxiv.org/abs/1409.4842`.

Saurabh Verma and Zhi-Li Zhang. Stability and generalization of graph convolutional neural networks. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, 2019.

Yewen Wang, Ziniu Hu, Yusong Ye, and Yizhou Sun. Demystifying graph neural network via graph filter assessment, 2020. URL `https://openreview.net/forum?id=r1erNxBtwr`.

B. Weisfeiler and A. Leman. A reduction of a graph to a canonical form and an algebra arising during this reduction (in russian). *Nauchno-Technicheskaya Informatsia*, 9, 01 1968.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2019.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2018a.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. *arXiv preprint arXiv:1806.03536*, 2018b.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2016.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. 2018.