Measuring the Expressiveness of GNNs

Sneha Silwal(ss14499) **Sriram Ramesh(sr5824)** **Xiang Pan(xp2030)**

1 Introduction

The question of whether Graph Neural Networks(GNNs) as described in the spatial and spectral domains are equally expressive leads to the more basic questions: what is the expressiveness of a GNN? How are the different ways we can measure this? And rather ambitiously, is there a common framework to view all these methods through? If not, how did these methods differ and is there value in having multiple approaches to measure expressiveness?

We start by introducing the problem statement and various aspects of graphs & graph networks to consider: the homophilious nature of some graphs, spectral perspectives for examining them, and desired invariant or equivariant properties of models.[1] We describe the most prevalent way of measuring expressiveness— the use of the Weisfeiler-Lehman graph isomorphism test and also share other methods related to it. One such method is to create universal approximation theorems for invariant and equivariant GNNs, which proves that they have equal expressive powers as continuous invariant and equivariant functions, respectively[1]. Another approach identifies the limits of WL approach in identifying substructures or patterns in graphs, and notes that by extending the scope of neighboring nodes which are examined (in a technique called relational pooling), different substructures which aren't able to be distinguished by the 1-WL or 2-WL levels can then be identified [2]. Instead of looking to UAT functions, yet another approach is to look at the logical formalisms and try to create a GNN which captures the class of functions covered by FOC_2 [3] .This is relevant to the WL test as prior work has shown that "k-variable language with counting is equivalent to the $(k-1)$-dimensional Weisfeiler-Lehman method." [4]

We then also explore some alternative perspectives with which to look at GNNs. From the spectral approach, one method finds the frequency response for each GNN method and explores what that can tell us about the graph's ability. [5] Atamna et. al. try to find a useful distance measure to capture the different representative power of two GNNs.[6] Lastly, we look to graph moments, Dehmamy et al bounded the ability of graph convolutional networks (i.e., GNN mpw/o messaging functions) to compute specific polynomial functions of the adjacency matrix. Their method is described as a "graph stethoscope" to detect the underlying processes at work in generating various graphs. [7]

2 Problem Statement

2.1 Graph

Let G be a graph with n nodes and an arbitrary number of edges. Adjacency matrix $A \in 0, 1^{n \times n}$ and features are defined on nodes by $X \in R^{nf_0}$, with f_0 the length of feature vectors. A graph Laplacian is $L = D - A$ (or $L = I - D^{-1/2}AD^{1/2}$) where $D \in R^{n \times n}$ is the diagonal degree matrix and I is the identity. Through eigendecomposition, L can be written by $L = Udiag(\lambda)U)^T$ where each column of $U \in R^{n \times n}$ is an eigenvector of L, $\lambda \in R^n$ gathers the eigenvalues of L and $diag(.)$ function creates a diagonal matrix whose diagonal elements are from a given vector. We use superscript to refer same kind variable as base. For instance, $H^{(l)} \in R^{nf_l}$ refers node representation

[1] check our blog here:
https://www.notion.so/Measuring-the-Expressiveness-of-GNNS-c83de0af20594a13a3b76adb22c1f70d

Preprint. Under review.

on layer l whose feature dimension is f_l. A Graph Convolution layer takes the node representation of the previous layer $H^{(l-1)}$ as input and produces a new representation $H^{(l)}$, with $H^{(0)} = X$.

2.2 Isomorphic graph and Non Isomorphic graph

Homophilous Graphs is defined as connected nodes tend to be similar with respect to a specific attribute. , e.g., interest in the social network and academic network.

Non-Homophilous Graphs: The definition is the opposite to the definition of Homophilous Graphs. There are some typical Non-Homophilous Graphs: gender relations in social or interaction networks and biological structures such as in food webs and protein interactions(PPI network)

2.3 Invariance and Equivariance in Graphs

Why we care about Invariance and Equivariance?

The graph is naturally disordered, when we care about the node characteristics, the node with its characteristics should not be changed with the reordering, in other words, the order of characteristics should be reordered as the number reorders, we need equivariance. When we care about graph level features, we want to change the node number without affecting the graph features, we need invariance. [8] names the graph permutation equivariance as Gobal Equivariance and the operator permutation invariance as local invariance.

Relation to expressive power

The ability to approximate invariant functions and equivariant functions can be seen as a expressive-power of GNN. Besides that, we can also use such perpoties to analyze the performance and guide us to design more powerful GNNs. Because of their scalability to large graphs, graph convolutional neural networks or message passing networks are widely used. However, such networks, which pass messages along the edges of the graph and aggregate them in a permutation invariant manner, are fundamentally limited in their expressivity [9].

2.4 Spectral GNNs

Initially, Graph Laplacian eigenvalues were interpreted as notions of frequency [10]. This helped in the spectral filtering in the frequency domain by transposing convultion theory to graph. The frequency can be defined by $x_{flt} = Udiag(\Phi(\lambda))Ux$, where $\Phi(.)$ is the desired filter function. So, a graph convolution layer in spectral domain can be written by a sum of filtered signals followed by an activation function as in [11]

$$H_j^{(l+1)} = \sigma\left(\sum_{i=1}^{f_l} U.diag(F_i(^{(l,j)})U^T H_i^{(l)}\right) \tag{1}$$

where σ is the activation function and $F^{(l,j)}$ is the weight vector

Drawbacks

- Matrix Multiplication of U and U^T for computing Fourier and Fourier transform
- Not a general approach as $F_i^{(l,j)}$ is composed of contribution from eigenvectors of Graph Laplacian

This limitations lead to the design $F_i^{(l,j)}$ as function of eigenvalues given as

$$F_i^{(l,j)} = B[W_{i,j}^{(l,1)} \dots W_{i,j}^{(l,s_e)}]$$

where $B \in R^{n,s_e}$, $s = 1 \dots s_e$ denotes the number of filters, and each column in B matrix is given $B_{k,s} = \Phi_s(\lambda_k)$ where $k = 1 \dots n$ denotes the index of eigenvalues.

2.5 Spatial GNNs

Spatial GNN aggregate the neighborhood nodes and updates the current node for the next layer.

$$H_{:v}^{(l+1)} = upd\left(g_0(H_{:v}^{(l)}, agg\left(g_1(H_{:u}^{(l)} : u \in \mathcal{N}(v)\right)\right) \tag{2}$$

where $\mathcal{N}$ is the set of neighboring nodes connected to the current node v and $g_0, g_1 : R^{n \times f_l} \rightarrow R^{n \times f_{l+1}}$ The choice of upd, agg, g_0, g_1 and $\mathcal{N}(v)$ determines the capability of the model.

2.6 Bridging Spatial and Spectral GNNs

Convolution support helps to express the contribution of neighbors for a particular node. Within this generalization, GNNs differ from each other by the choice of convolution supports $C(s)$. Spectral GNN (1) is special case of general equation (3) with convolution kernel set as $C^{(s)} = Udiag(\Phi_s(\lambda))U^T$ [5].

$$H^{(l+1)} = \sigma\left(\sum_s C^{(s)} H^{(l)} W^{(l,s)}\right) \tag{3}$$

where $C(s) \in R^{n \times n}$ is the s^{th} convolution support, $W^{(l,s)} \in R^{f_l \times f_{l+1}}$ is the trainable matrix for the l^{th} layer and s^{th} filter.

In case of spatial, we have $upd(x, y) = \sigma(x + y)$, agg is a sum (or weighted sum) of the defined neighborhood nodes contributions and gi applies linear transformation, one can trivially show that mentioned spatial GNNs (2) can be generalized as propagation of the node features to the neighboring nodes followed by feature transformation and activation function of the form (3)

A Trainable-support is a Graph Convolution Support $C^{(s)}$ with at least one trainable parameter that can be tuned during training. If C (s) has no trainable parameters, i.e. when the supports are pre-designed, it is called a fixed-support graph convolution. Formally, we can define a trainable support by:

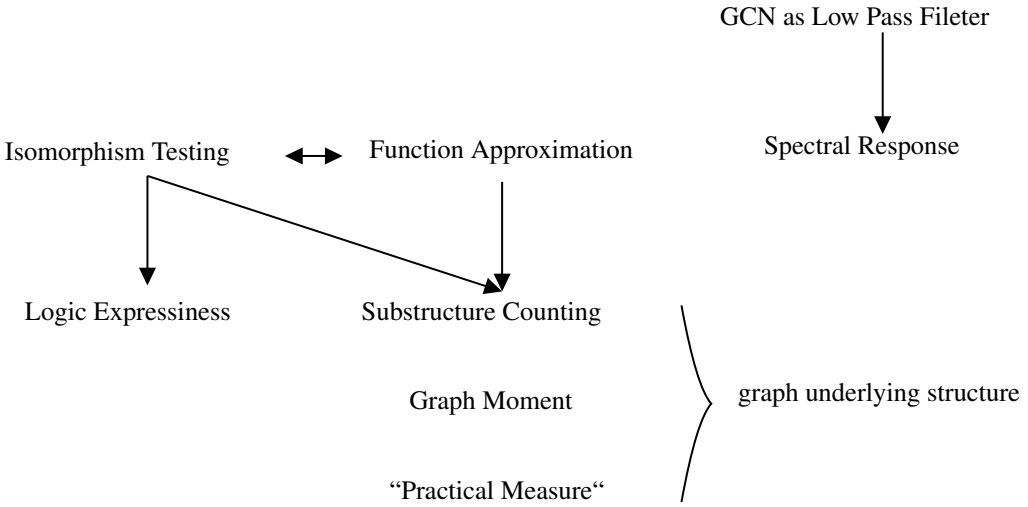
$$(C^{(l,s)})_{(v,u)} = h_{s,l}(H^{(l)}_{:v}, H^{(l)}_{:u}, E^{(l)}_{v,u}, A)$$

where $C^{(l,s)}$ is s^{th} convolution support of l^{th} layer $E^{(l)}_{v,u}$ shows edge features on layer l from node v to node u $h(.)$ is trainable model parameterized by s, l

From another way, convolution support can be a function or constant depending on the definition of GNN.[12] constructs the mapping between spatial and spectral domain.

3 Expressive Power Definition

In this part, we are discussing different definitions. The expressiveness is quite related to the practical tasks, it measures the ability of GNN in one specific representation task and its related real word task. In the meanwhile, they are related to each other:



3.1 Definition 1:Graph Isomorphism Tests

Weisfeiler-Lehman (WL) Graph Isomorphism Test [13] is widely used to distinguish a broad class of graphs. It is a reduction of the Graph to a canonical form which helps in solving graph-theoretic problems.

Algorithm

1. For iteration i of the algorithm we assign to each node a tuple $L_{i,n}$containing the node's old compressed label and a multiset of the node's neighbors' compressed labels. A multiset is a set (a collection of elements where order is not important) where elements may appear multiple times [14] .
2. At each iteration we will additionally be assigning to each node a new "compressed" $C_{i,n}$ label for that node's set of labels. Any two nodes with the same $L_{i,n}$ will get the same compressed label.

[9] proposes graph isomorphism tests as a characterization of the power of graph neural networks. They show that if a GNN follows a neighborhood aggregation scheme, then it cannot distinguish pairs of non-isomorphic graphs that the 1-WL test fails to distinguish. Therefore this class of GNNs is at most as powerful as the 1-WL test.

[15] establishes the equivalence of anonymous GNN mp (those that do not rely on node identification) to the Weisfeiler-Lehman (WL) graph isomorphism test: The equivalence implies that anonymous networks are blind to the many graph properties that WL cannot see: e.g., any two regular graphs with the same number of nodes are identical from the perspective of the WL test.

3.2 Definition 2: Functions Approximation

The universal approximation theory applied to neural networks allows us to approximate it using polynomials, and the same can be done for GNNs. [16] proposes a specific neural network architecture with polynomial layers can achieve universal approximation of invariant or equivariant functions. In graph structure, we are more concerned about the invariant function and equivariant properties. Invariance to permutation means that the output will not change regardless of permutations to the input, while equivariance means the output must be permuted when the input is permuted. [17] constructs the proof on a basis of invariant polynomials and on classical universality of MLPs, but it is not practical in high order structure like graphs. Maron et al. proposes G-equivariant version of universal approximation theorem: the universal approximation of G-invariant networks, which are constructed based on the linear invariant and equivariant layers [18]. If the order of the tensor involved in the networks can grow as the graph gets larger, it is not practical in GNNs setting due to the dependence on the graph size.

Keriven & Peyré [1] use the Stone-Weierstrass theorem for algebras of real-valued functions extend to the equivariant functions. They consider GNNs of the form

$$f(G) = \Sigma_{s=1}^{S} Hs[\rho(Fs[G] + Bs)] + b_i \tag{4}$$

F_s is a linear invariant function, $Fs : R^{n^d} \to R^{n^{k_s}}$ and Hs are invariant linear operators $H : R^{n^{k_s}} \to R$, Bs is the bias term s $Bs \in R^{n^{k_s}}$ are equivariant, so that $Bs = \sigma(\star)Bs$ for all σ, and ρ is any Lipschitz pointwise non-linearity function.

The Stone-Weierstrass theorem allows one to define a sub-algebra using their definition for GNNs to describe functions which capture the properties of graph invariance (G_inv).

First they describe equivalence classes for graphs, which for each class defines the set of graphs which are isomorphic to each other. For proof of their theorem they show 1) that the GNNs defined by function f forms a subalgebra by closing it under multiplication, 2) show that the set of graphs graphs described as invariant and are defined as such by the edit distance form a compact set, and 3) the function f maps to separable points. From this they conclude that for any G_{inv} class, it can be expressed by some degree of accuracy by a one layer invariant Neural Network.

Since MPNNs are widely used versions of GNNs and its formulation is "similar" to the WL test, Chen et al.[19] analyzes the MPNN by UAT:

$\mathcal{F}$ is **universally approximating**:if $\forall$ function g: $\mathcal{G} \to \mathbb{R}$ and $\forall \epsilon > 0, \exists f \in \mathcal{F}$, such that $\sup G \in \mathcal{G} |g(G) - f(G)| < \epsilon$

$\mathcal{F}$ is **GISO-discriminating**: if $\forall G1, G_2 \in \mathcal{G}$ that are not isomorphic, $\exists f \in \mathcal{F}$ such that $f(G_1) \neq f(G_2)$

If $\mathcal{F}$ is Universally approximating on $\mathcal{G}$, then it is also GISO-discriminating on $\mathcal{G}$; If $\mathcal{F}$ is GISO-Discriminating on $\mathcal{G}$, then $\mathcal{F}^{+1}$ is universally approximating on $\mathcal{G}$. If $\mathcal{F}$ is universally approximating on $\mathcal{G}$, then it is also GISO-discriminating on $\mathcal{G}$; If $\mathcal{F}$ is GISO-discriminating on $\mathcal{G}$, then $\mathcal{F}^{+1}$ is universally approximating on $\mathcal{G}$. MPNNs are not fully GISO-discriminating, thus MPNN are not universally approximating. However, MPNN is widely used in many tasks, which suggests that the UAT may not be an approrit measure for some specific tasks.

3.3 Definition 3: Count Substructure

Morris et al. [20] states that 1-WL GNN are not capable of capturing simple graph theoretic properties, e.g., triangle counts, which are an important measure in social network analysis. This is the motivation to examine beyond a node's immediate neighbors and to look at the graph's substructure.

GNNExplainers[21] attempt to find the nodes that contribute to the outputted prediction. Given a GNN model and its predictions, it produces a subgraph of the input graph and a subset of node features that contributed to the prediction. They note that related work on interpretability can be grouped into two groups: the first method is to"formulate simple proxy models of full neural networks", the second is to "identify important aspects of the computation." This is yet another motivation, for one to find substructures of graphs that contribute to certain outputs.

Chen et al.[2] proposes that **substructure counting** task: First, they define the subgraph and induced subgraph: Given two graphs $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B) := G_A$ is a subgraph of G_B if $V_A \subseteq V_B$ and $E_A \subseteq E_B$ G_A is an induced subgraph of G_B if $V_A \subseteq V_B$ and $E_A = E_B \cap (V_A \times V_A)$.

Based on subgraph and induced-subgraph definition, they defines the task as: Let G be a graph, and $G^{[\mathrm{P}]}$ be a pattern we are interested in counting. The subgraph-count of $G^{[\mathrm{P}]}$ in G, denoted by $\mathrm{C}_S\left(G; G^{[\mathrm{P}]}\right)$, is the number of subgraphs of G that are isomorphic to $G^{[\mathrm{P}]}$. The induced-subgraph-count of $G^{[\mathrm{P}]}$ in G, denoted by $\mathrm{C}_l\left(G; G^{[\mathrm{P}]}\right)$, is the number of induced subgraphs of G that are isomorphic to $G^{[\mathrm{P}]}$.

Link to Graph Isomorphism Testing and Function Approximation $\mathcal{F}$ is a family of GNNs, $G^{[\mathrm{P}]}$ is a given pattern. We can define the substructure task as Function Approximation: Can $\mathcal{F}$ approximate $\mathrm{C}_S\left(\cdot; G^{[\mathrm{P}]}\right)$ or $\mathrm{C}_l\left(\cdot; G^{[\mathrm{P}]}\right)$ as functions on the graph space arbitrarily well? In the meantime, we can define the from the graph isomorphism testing perspective: Given any two graphs, $G^{[1]}$ and $G^{[2]}$ with different (induced-)subgraph-counts of a pattern $G^{[\mathrm{P}]}$, can we always find a function in $\mathcal{F}$ that distinguishes them?

Based on the inexpressivenss of the task result and task is directly relavant to many real word problems, [22] proposes topologically-aware message passing scheme based on substructure encoding.

3.4 Definition 4: Logic Expressiveness

For binary node classification tasks, we can look to logical formalisms and attempt to characterize the set of logical statements that are captured by aggregate-combine GNNs (generic GNNs) [3]. Barceló et al. introduces this idea and sets out to create a GNN model that extends AC-GNNs and captures the whole class of FOC_2 statements

FOC_2 is described as there are at least N nodes satisfying some function ϕ. This measurement is related to the WL-test in that "two nodes in a graph are classified the same by the WL test if and only if they satisfy exactly the same unary FOC2 formulas."

They acknowledge that there is already a link between the expressive power of AC-GNNs and the WL test, and that there is also some sort of equivalence between the WL test and FOC_2, but this does not imply a link between AC-GNNs and FOC_2. They extend AC-GNN by adding a global vector and

a readout function:

$$\boldsymbol{x}_v^{(i)} = \mathrm{COM}^{(i)}\left(\boldsymbol{x}_v^{(i-1)}, \mathrm{AGG}^{(i)}\left(\left\{\boldsymbol{x}_u^{(i-1)} \mid u \in \mathcal{N}_G(v)\right\}\right), \mathrm{READ}^{(i)}\left(\left\{\boldsymbol{x}_u^{(i-1)} \mid u \in G\right\}\right)\right) \tag{5}$$

They present a theorem that every FOC_2 classifier can be captured by a simple homogenous ACR-GNN. On their experiments, this new ACR-GNN had improved performance on tasks that AC-GNNs were unable to perform well on. These examples were Erdős–Rényi graphs with one single blue node and also more complex classifiers. They also tested both models on Protein-Protein Interaction (PPI) benchmark and the performance of the two were comparable. They mention future work, where they hope to use their results to extract which logical formalisms are expressed by the GNN. This would help with regards to the explainability of the GNN models.

3.5 Definition 5: Spectral Measurements

GNN expressive power has been associated with their capability to distinguish 2 graphs are isomorphic or not. Graph isomorphism is NP-intermediate and WL test gives sufficient evidence in polynomial time. Most of the expressive power of graph has been related to their performance in WL test and classified as 1-WL, 2-WL. This does not explain the success rate of GNN on benchmarks datasets and cannot distinguish between two methods with same WL order.

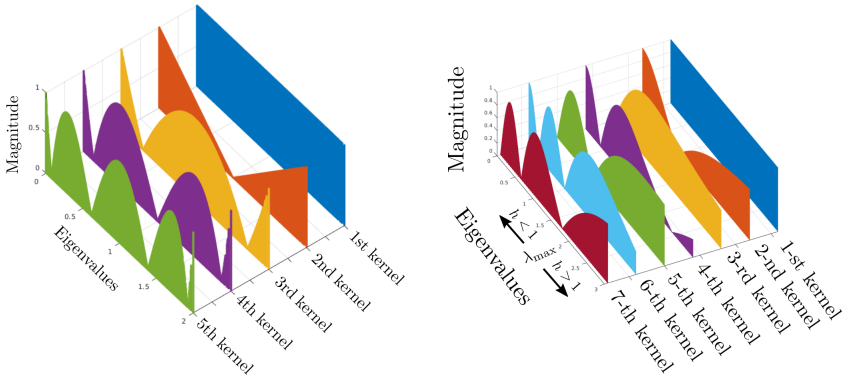
MPNNs and GraphNets are the only attempts to merge spectral and spatial based approaches. These models fails to generalize custom designed spectral filters.

Spectral-designed have supports as function of eigenvalues and eigenvectors of L while others are spatially designed.Frequency Profile for $C^{(s)}$ in spectral domain can be defined as: $\Phi_s(\lambda) = diag^{-1}(U^T C^{(s)} U)$ where $diag^{-1}$ returns vector of diagonal elements and U is eigenvector of L. In Spatial, we compute full frequency profile as they don't have a well defined convolution kernel $\Phi_s(\lambda) = U^T C^{(s)} U$. We use the frequency profile to study frequency response of different types of GNN.

Table 1: Summary of the studied GNN models.

	Design	Support Type	Convolution Matrix	Frequency Response
MLP	Spectral	Fixed	$C = I$	$\Phi(\boldsymbol{\lambda}) = \mathbf{1}$
GCN	Spatial	Fixed	$C = \tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}$	$\Phi(\boldsymbol{\lambda}) \approx \mathbf{1} - \boldsymbol{\lambda}\overline{p}/(\overline{p}+1)$
GIN	Spatial	Trainable	$C = A + (1+\epsilon)I$	$\Phi(\boldsymbol{\lambda}) \approx \overline{p}\left(\frac{1+\epsilon}{\overline{p}} + \mathbf{1} - \boldsymbol{\lambda}\right)$
GAT	Spatial	Trainable	$C^{(s)}_{v,u} = e_{v,u} / \sum_{k \in \tilde{\mathcal{N}}(v)} e_{v,k}$	NA
CayleyNet[a]	Spectral	Trainable	$C^{(1)} = I$ $C^{(2r)} = Re(\rho(hL)^r)$ $C^{(2r+1)} = Re(\mathbf{i}\rho(hL)^r)$	$\Phi_1(\boldsymbol{\lambda}) = \mathbf{1}$ $\Phi_{2r}(\boldsymbol{\lambda}) = \cos(r\theta(h\boldsymbol{\lambda}))$ $\Phi_{2r+1}(\boldsymbol{\lambda}) = -\sin(r\theta(h\boldsymbol{\lambda}))$
ChebNet	Spectral	Fixed	$C^{(1)} = I$ $C^{(2)} = 2L/\lambda_{\max} - I$ $C^{(s)} = 2C^{(2)}C^{(s-1)} - C^{(s-2)}$	$\Phi_1(\boldsymbol{\lambda}) = \mathbf{1}$ $\Phi_2(\boldsymbol{\lambda}) = 2\boldsymbol{\lambda}/\lambda_{\max} - \mathbf{1}$ $\Phi_s(\boldsymbol{\lambda}) = 2\Phi_2(\boldsymbol{\lambda})\Phi_{s-1}(\boldsymbol{\lambda}) - \Phi_{s-2}(\boldsymbol{\lambda})$

[a] $\rho(x) = (x - \mathbf{i}I)/(x + \mathbf{i}I)$



(a) First 5 ChebNet supports

(b) First 7 CayleyNet supports

Figure 1: Frequency profiles ($\Phi_s(\boldsymbol{\lambda})$)

For ex-

ample, the theoretical frequency response of each support of ChebNet can be defined as $\Phi_1(\lambda) = 1, \Phi_2(\lambda) = 2\lambda/\lambda_{max} - 1, \Phi_k(\lambda) = 2\Phi_2(\lambda)\Phi_{k-1}(\lambda) - \Phi_{k-2}(\lambda)$ As we can see from the graph first kernel is an all-pass filter, second kernel is low-pass and high-pass filter for ChebNet. Using the frequency response of different GNN, we can compare their performance and answer questions on suitability of low-pass GNNs for semi-supervised learning and ability of GNN to learn a type of filter.

[23] shows that a GCN approximately works as a low-pass filter plus an MLP in a certain setting. Although they analyzed finite-depth GCNs, our theory has similar spirits with theirs because our "information-less" space corresponds to the lowest frequency of a graph Laplacian. Another point is that they imposed assumptions that input signals consist of low-frequent true signals and high-frequent noise, whereas we need not such an assumption.

3.6 Definition 6: "Practical Measure"

A simple, straightforward way to measure the expressiveness of a graph would be to have some number for the representation power of the model which would indicate the graphs performance on tasks. Antamna et al.[6] attempted to come up with a measure that would capture the expressiveness of a graph. Their line of reasoning was that expressiveness describes the ability to map different graphs to different representations. They wanted to test whether higher representative measures would result in greater accuracy on tasks. If successful, it would be helpful to have a "representative" score to predict how a graph would perform on future tasks. Their measure of expressiveness for a GNN $\mathcal{A}$:

$$\mathcal{E}(\mathcal{A}) = mean(\{||z_{G_i} - z_{G_j}||_2 : i, j = 1, 2...m, i \neq j\})\mathcal{E}(\mathcal{A}) \tag{6}$$

$\mathcal{E}(\mathcal{A})$ is the average distance between the node embeddings for two graphs, G_i and G_i. They trained their model, SPI-GCN using a loss function that maximized the measure of expressiveness. However, their tests showed that the accuracy of their was not better than other GNN methods such as D_{GCNN}, PSCN, GIN. They also tested various versions of SPI-GCN with different penalty factors for the expressiveness. Their results did not show any relation between the expressiveness and the accuracy of the models, suggesting that the constructed expressiveness measure is not sufficient to predicting a graphs performance on a task.

3.7 Definition 7: Graph Moments

Dehmamy et al. investigate the representative power of GNNs through graph moments[7]. They point out that although GCNs are used as feature extractors for graph data, we have yet to identify to what extent GCNs can capture graph features. Their work sets out to "analyze the representation power of GCNs in learning graph topology using graph moments, capturing key features of the underlying random process from which a graph is produced."

They review the graph moments that are able to be represented by GNNs and define the connections between the GCN properties (depth/width) and it's representational power.

Graph moments "characterize the random process from which the graph is generated." A p^{th} order moment is defined as follows:

$$M_p(A) = \Pi_{q=1}^{p}(A \cdot W_q + B_q) \tag{7}$$

Graph moment will be composed of an ensemble average of an order p polynomial of A. The graph moment relates to the graph's Wiener Index and degree distance which are useful when considering the structure of molecules in chemistry [24]. Graph moments also help with graph color and Hamiltonicity by encoding information about the structure of a graph. [7]

Graph moments are node permutation invariant. The information captured– distribution of degrees, paths of a given length, number of triangles, etc does not change when the graph is relabelled.

Dehmamy et al. generate a "modular" GCN which is composed of three functions, A (adjacency matrix), $D^{-1}A$ and $D^{-1/2}AD^{-1/2}$. Their model also has a residual layer which has concatenations from each layer of the model. They include these three modules, explaining that different propagation rules are able to learn different types of moments.

For their experiments, the generate graphs using the Barabasi-Albert model, the Erdos-Renyi model, and a "fake" BA, and they show that they are able to discriminate between BA and ER graphs. They attribute this success to having different propagation rules in their modules.

Graphs invariants based on the distance between vertices are used in chemistry since there are correlations between these measures and the molecules' other properties. [24]. Graph moments were introduced to generalize all of these invariants, that is, given a graph moment we are then able to derive these invariants. That means there may be other properties of these graphs which we are interested in, rather than just whether they are isomorphic to one another. These properties are captured by the p-moment, and the paper by Dalfo presents a method so that these non-isomorphic graphs will have the same p-moments for every p.

4 Discussion

In addition to the expressive power of GNN models, there are other factors may affect the performance of GNNs on the target tasks.

Training Limitation [25] illustates that GNNs may be limited by the training strategy, e.g. SGD, the theoretical proof can not guarantee that GNNs with the expressive power can solve any that kind of problems (related-work).

Node Level Information Loss [26] shows that when its weights satisfy the conditions determined by the spectra of the (augmented) normalized Laplacian, its output exponentially approaches the set of signals that carry information of the connected components and node degrees only for distinguishing nodes. However, for GNNs, some paper [27, 28] have illustrated that node representations go indistinguishable (known as over-smoothing) and prediction performances severely degrade when we stack many layers.

Generated Datasets Many of the above works which studied the expressiveness of GNNs had experiments run on generated graph datasets. These generated datasets may contain regular pattern and less than the real world test. It's left to see how these models will perform on collected datasets and real world tasks.

Theoretical Results Additionally, many of the theoretical ideas presented dealt with the case of a 1 hidden-layer GNN or simplified the parameters to the case of examining 1 neighbor or concatenating the representation of nodes. For the case of the theoretical results, future work must focus on extending the results to larger GNNs. On the experimentation side, there is future work to extend these results to more complex examples; one such example is to examine more complex substructures than those within l=1.

Possible way to improve Equivariance GNNs [8] suggests that the local invariance is not necessary to global equivariance. We can directly construct an equivariance GNNs but it is more on the theoretical area[29, 18]. [30] introduces new local invariant aggregation operator, while [8] introduces some local equivariant aggregator.

5 Conclusion

We've reviewed seven different ways people have approached expressiveness. The work by Xu. [9] asserts that the expressiveness of a GNN is characterized by its ability "in learning to represent and distinguish between different graph structures."

In most cases, the discriminative power of WL test is sufficient given the model's objective. But, the task of representing and distinguishing graph structures is difficult, and there are many different aspects of a graph that one may be interested in, dependent on the task at hand. Even the WL-test has its limitations as there are trade-offs to its property of node-invariance and the aggregation of immediate neighbors.

An observation to be made here is that the methods such as ACR-GNN (used to capture the class FOC_2) and the graph stethoscope (to capture graph moments) added non-local information through the introduction of either global vectors or residual layers. The LRP model (for counting substructures) examined a set of nodes distance-l for every vertex. The tasks these different models set out to do also required knowledge of the unique properties and characteristics of the graph that are not

immediately clear from any node's immediate neighbors. In those instances, if one is interested in node-classification tasks or more specific characteristics of a graph such as the presence of specific substructures or the underlying properties of the graphs topologies, exploring these alternative methods for examining expressiveness would be useful.

It's clear that there are trade-offs in each approach, and the method one may be interested in depends on the context in which graphs are studied. Expressiveness, a broadly defined term, has various aspects to it and the relevant definition is dependent on the task. It can be the common lens through which we analyze various GNNs and their performance on respective tasks.

References

[1] Nicolas Keriven and Gabriel Peyré. Universal Invariant and Equivariant Graph Neural Networks.

[2] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can Graph Neural Networks Count Substructures? 2020.

[3] Pablo Barcelo, Jorge Perez, Egor V Kostylev, and Juan Reutter. The Logical Expressiveness of Graph Neural Networks. page 21.

[4] Jin-yi Cai, Martin Furer, and Neil Immerman. An Optimal Lower Bound on the Number of Variables for Graph Identification. page 22.

[5] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. 2021.

[6] Asma Atamna, Nataliya Sokolovska, and Jean-Claude Crivello. A principled approach to analyze expressiveness and accuracy of graph neural networks. In Michael R. Berthold, Ad Feelders, and Georg Krempl, editors, *Advances in intelligent data analysis XVIII*, pages 27–39, Cham, 2020. Springer International Publishing.

[7] Nima Dehmamy, Albert-László Barabási, and Rose Yu. Understanding the Representation Power of Graph Neural Networks in Learning Graph Topology.

[8] Pim de Haan, Taco S Cohen, and Max Welling. Natural graph networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in neural information processing systems*, volume 33, pages 3636–3646. Curran Associates, Inc., 2020.

[9] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks?

[10] David I. Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains. *IEEE Signal Processing Magazine*, 30(3):83–98, May 2013.

[11] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[12] Zhiqian Chen, Fanglan Chen, Lei Zhang, Taoran Ji, Kaiqun Fu, Liang Zhao, Feng Chen, and Chang-Tien Lu. Bridging the Gap between Spatial and Spectral Domains: A Survey on Graph Neural Networks. *arXiv:2002.11867 [cs, stat]*, June 2020.

[13] AA Leman and B Weisfeiler. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, 2(9):12–16, 1968.

[14] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[15] 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. *arXiv:1810.02244 [cs, stat]*, February 2020.

[16] Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *arXiv:1804.10306 [cs]*, April 2018.

[17] Akiyoshi Sannai, Yuuki Takai, and Matthieu Cordonnier. Universal approximations of permutation invariant/equivariant functions by deep neural networks.

[18] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *CoRR*, abs/1812.09902, 2018.

[19] Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with GNNs. *arXiv:1905.12560 [cs, stat]*, May 2019.

[20] 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. *arXiv:1810.02244 [cs, stat]*, February 2020.

[21] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNN explainer: A tool for post-hoc explanation of graph neural networks. *CoRR*, abs/1903.03894, 2019.

[22] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting. *arXiv:2006.09252 [cs, stat]*, January 2021.

[23] Hoang NT and Takanori Maehara. Revisiting Graph Neural Networks: All We Have is Low-Pass Filters, 2019.

[24] C. Dalfó, M. A. Fiol, and E. Garriga. Moments in graphs. 2012.

[25] Andreas Loukas. What graph neural networks cannot learn: Depth vs width, 2020.

[26] Kenta Oono and Taiji Suzuki. Graph Neural Networks Exponentially Lose Expressive Power For Node Classification. page 37.

[27] 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. *arXiv:1909.03211 [cs, stat]*, November 2019.

[28] Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 338–348, 2020.

[29] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in neural information processing systems*, volume 32. Curran Associates, Inc., 2019.

[30] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. DeeperGCN: All You Need to Train Deeper GCNs. *arXiv:2006.07739 [cs, stat]*, June 2020.