

Graph Rhythm Network: Beyond Energy Modeling for Deep Graph Neural Networks

Yufei Jin and Xingquan Zhu

Department of Electrical Engineering and Computer Science, Florida Atlantic University
Boca Raton, FL 33431, USA
{yjin2021, xzhu3}@fau.edu

Abstract—Graph neural networks (GNN) have been commonly used for learning and classifying objects with correlated relationships. To date, many GNN architectures exist, but majority of them only work well on shallow networks due to the oversmoothing phenomenon, where node features become similar to each other, as the layer increases. In this paper, we point out that the key to create an informative deep GNN is to have an adaptive *feature updating rate* control for each node, where the updating rate should take each node’s locality into consideration through shared trainable weight parameters. Accordingly, we advocate a new *graph rhythm* modeling as a generalized mechanism to the Dirichlet energy based approaches. Instead of merely modeling difference between nodes, like Dirichlet energy based approach does, graph rhythm focuses on omni-directional relationship mapping between each node and its neighbors. Such a mechanism provides a more general ways of capturing patterns between nodes (*i.e.* graph rhythm) for effective graph neural network learning. Experiments and comparisons, demonstrate the performance gain and show that GRN can help create GNNs with deep layers, without suffering from performance deterioration or having better performance than shallow networks.

Index Terms—Graph rhythm, graph neural network, oversmoothing, graph embedding

I. INTRODUCTION

Graph neural networks (GNN) have been commonly used in many machine learning tasks involving networked objects, such as network node classification, link prediction, chemical compound classification [1], and recently graph distribution learning [2], *etc.*. To take network topology into consideration for learning, a series of works, such as Graph Convolution Network (GCN) [3], Graph Attention Network (GAT) [4], and GraphSage model [5] have been proposed to solve respective challenges. Despite of unique motivations, a key component of these networks is to find proper aggregation mechanism for feature updating. Aggregation is naturally required by the permutation invariant properties of the graph data, but it often results in performance deterioration for deep layers. As layer increases, embedding features learned from the network tend to collapse to similar values, indicating node/topology information loss through aggregation operation, a phenomenon commonly referred to as oversmoothing [6].

A. Deep GNN vs. Oversmoothing

Existing methods alleviating oversmoothing in graph neural networks largely fall into five main categories [6]: residual-based, random-masked based, diffusion-based, transformer-based, and energy-based approaches. Early works include a

series of dropping mechanisms attempting to mask part of node features and/or graph topology. A representative work is DropMessage [7], which alleviates oversmoothing up to several layers but unsuccessful for deeper layers. This part of work also does not show clear evidence of performance gain with layer increases.

To tackle oversmoothing, Residual-based Message Passing Neural Network (MPNN) is commonly used as a graph neural network backbone. Transformer-based approaches introduce attention-mechanism or learn new topology induced from features. Such an attention mechanism typically introduces high computation costs and complexities, and there is limited evidence showing that it can help train deep graph neural networks. Diffusion-based models leverage continuous Ordinary/Partial Differential Equations (ODE/PDE) and their discretizations, which often lead to residual-based MPNN structures.

For message passing neural networks, a critical component is its residual-based approach, an idea borrowed from computer vision field which has shown to alleviate the oversmoothing in deep neural networks, as demonstrated in GCNII [8]. The key idea behind the residual-based message passing is to balance the updated feature by mixing information aggregated from neighbors and the original features. Despite its success, existing methods still do not show good performance as layers become sufficiently large, such as over 10 layers or more, mainly because learning fails to gain additional benefits from deep depth structures. This is mainly attributed to the challenge of solving a double-sided problem: Simply trying to learn unique embedding features for each node (*i.e.* pushing nodes away from each other) does not necessarily deliver good solutions to resolve the oversmoothing problem because such unique embedding features may not faithfully represent topology/content features of each node (*i.e.* a node expressive power issue), and therefore result in poor performance in downstream tasks [9].

Another line of works focus on defining a measure to reflect the extent of oversmoothing over layers and try to preserve the measure to some threshold. A popular measure for this line of works is called Dirichlet energy, which is the summation of nodes distance towards its one-hop neighbors. EGNN [10], for example, selects the coefficient for the residual component based on the lower bound of the energy. G2-gating [11], directly uses a $\tanh()$ mapping to control feature updating

rate through Dirichlet energy. It is observed that G2-gating achieves both deep layers without oversmoothing and can actually benefit from such deep structures with performance gain.

From the existing studies, there is a gap between alleviating oversmoothing *vs.* achieving additional performance gain from deep layers. In this paper, we intend to close the gap by delivering a solution to not only alleviate oversmoothing but also achieve better performance from deep layers.

B. Motivation and Contributions

In summary, existing research has demonstrated that deep graph structures often do not work better than a simple shallow version even if the Dirichlet energy is well preserved. This suggests that simply alleviating oversmoothing is insufficient, and there is a need to ensure embedding features can faithfully preserve content and topology information (*i.e.* node expressive power) for performance gain.

Motivated by the above observations, we propose to learn deep graph neural networks without performance degradation as layers become very deep. Our main theme is to establish a graph rhythm measure, which replaces Dirichlet energy, for feature updating. A unique strength of graph rhythm stems from its *omni-directional relationship mapping* nature, which is capable of modeling much boarder node relationships than Dirichlet energy, which is only limited to modeling node differences. Combining graph rhythm and residual message passing, a graph rhythm network (GRN) is proposed. Empirical comparisons demonstrate GRN's performance in creating very deep GNNs (*e.g.* networks with over 100 layers).

II. PROBLEM DEFINITION & PRELIMINARY

A. Symbols and Notations

A graph, denoted by $G = (V, E, X, Y)$, consists of a node set V with n nodes, an edge set E , a feature matrix $X \in \mathbb{R}^{n \times m}$ with m features for each node v_i , and a label set Y with labels being assigned to some nodes in V . Denote $A \in \mathbb{I}^{n \times n}$ the adjacency matrix of the graph G , with $A[i, j]=1$ if an edge exists between v_i and v_j , or 0 otherwise. For a node $v_i \in V$, we denote $x_i \in \mathbb{R}^{1 \times m}$ as v_i 's feature vector. Δ_{v_i} denotes the set of 1-hop neighbors of node v_i , and $\Delta_{v_i}^x$ denotes the feature value set for v_i 's 1-hop neighbors. Denote $F_\Theta(X, A)$ as an arbitrary graph coupled function or graph neural networks, *e.g.*, Graph Convolution, Graph Attention, or GraphSage. Denote l the l^{th} layer of the graph learning model on the graph, and $X^l \in \mathbb{R}^{n \times c}$ denotes the node embeddings learned at the l^{th} layer.

For ease of representation, we use X to represent both original feature space and the learned embedding feature space, because original node features are considered the initiative values of embedding features. In our derivations, we consider that original feature space dimension m and the embedding feature space c are equal (with $c = m$), because we can employ a projection layer with weight parameters $W_0 \in \mathbb{R}^{m \times c}$ to convert original node features to X^0 , as shown in Fig. 1.

B. Problem Definition

Given a graph G and a graph enabled neural network (*i.e.*, a Graph-coupled function $F_\Theta(\cdot)$) [12], our **goal** is to propose a generic design for creating deep graph neural networks whose performance will not decrease as the layer l increases, for at least $l \geq 20$. This goal will help advance the graph neural networks from shallow layers to be very deep, *e.g.* the performance of most existing GNNs, such as GCN or dropout-GCN [13], will deteriorate significantly as layer l increases.

In our research, we propose a graph rhythm as a new way of information aggregation between each node and its neighbors. By combing graph rhythm with message passing neural networks (MPNN), our proposed design can deliver GNNs with over 100 layers, but still achieve noticeable performance gain with peak performance at 64 layers.

C. Preliminary: Oversmoothing

Oversmoothing problem is a main challenge preventing a graph neural network architecture from being deep. Due to oversmoothing, the learned node embeddings tend to collapse to the same point (or a ball with small radius), as the number of layers increase for popular MPNN models. Several theorems, such as subspace theorem [14], ODE theorem [12], *etc.*, have been proposed to explain this phenomenon. Among them, Dirichlet energy is a core concept commonly used to explicitly quantify oversmoothing. Many methods, therefore, propose to tackle oversmoothing by taking Dirichlet energy into consideration during modeling process, either implicitly or explicitly.

1) *Dirichlet Energy*: Dirichlet energy measures total distance between a central node v_i and its 1-hop neighbors:

$$\varepsilon_{\text{DE}}(x^l) = \frac{1}{n} \sum_{i=1}^n \sum_{x_b \in \Delta_{v_i}^x} \|x_i^l - x_b^l\|_2^2 \quad (1)$$

Oversmoothing occurs when $\varepsilon_{\text{DE}}(x^l)$ decays exponentially as the layer increases l [12].

a) *local vs. global*: When using Dirichlet energy to guide embedding learning, most existing methods employ a local energy based mechanisms, as defined in Eq. (2).

$$\varepsilon_{\text{DE}}(x_i^l) = \sum_{x_b \in \Delta_{v_i}^x} \|x_i^l - x_b^l\|_2^2 \quad (2)$$

Being local, node feature updating only accounts for each node v_i 's own energy, instead of considering energy of other nodes. For example, in G2-gating [11], coefficient of the updated node is controlled by the Dirichlet energy. This helps G2-gating prevents all nodes locally converging to its smooth version (aggregation of its neighbors). Nevertheless, since local Dirichlet energy only accounts for node differences, as the local Dirichlet energy increases, the mapping through the $\tanh(\cdot)$ will monotonically increase and will only push the central node away from the smooth version but cannot consider more complexity relationships between nodes. As a result, G2-gating still falls short to leverage local energy for updating.

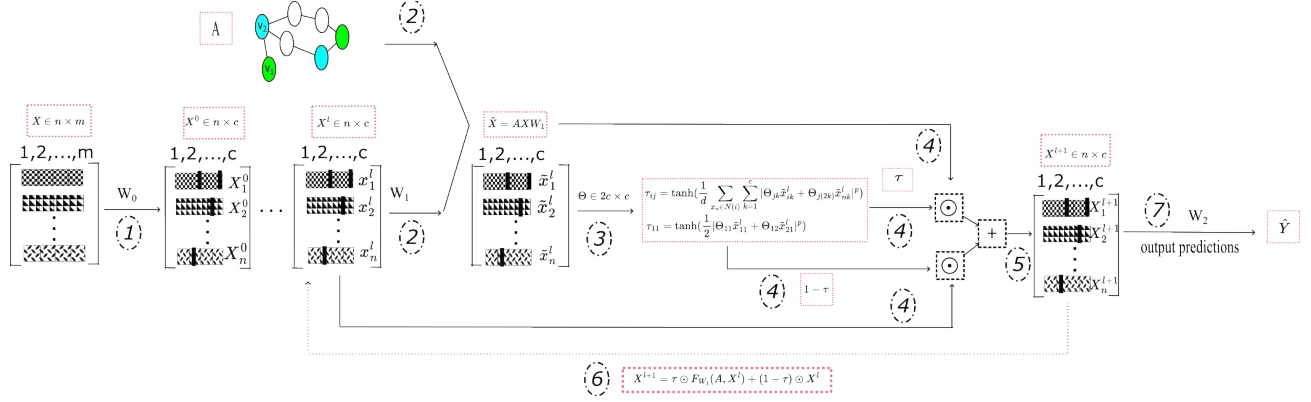


Fig. 1. The framework of the proposed graph rhythm network (GRN). From left to right: ① given a graph denoted by adjacency matrix A and feature matrix X , W_0 is first used to project each node's feature to a new embedding dimension c ; At any layer l , ② each node aggregate information from neighbors; ③ the aggregated information is used to calculate updating coefficient τ for each node; ④ and ⑤ carry out message passing to learn embedding for next layer X^{l+1} ; ⑥ iteratively repeating the process to the next layer; and ⑦ using learned embedding features for downstream tasks.

The limitation of lacking global view and new measure mechanism are the major hurdle preventing existing GNNs architectures from achieving very deep layers with better performance. Nodes in a graph have complex relationships (long-term and/or short-term relation). Instead of simply relying on node difference, like Dirichlet energy does, we need a more generalized form to model node relationships and an adaptive mechanism to learn to capture richer patterns. Our research provides direct answers to these questions.

III. METHOD

In this section, we first introduce technical details of the message passing neural networks (MPNN), which will serve as the basis for us to create graph rhythm as a new way of modeling node relationships. The remaining subsections will detail the proposed graph rhythm network (GRN).

A. Message Passing Neural Network (MPNN)

Residual message passing can be considered as a discretization of an ODE and is often used to help create deep neural networks [12] [11]. In the context of graph learning, assume X^l and X^{l+1} are embedding features learned from two consecutive layers l to $l+1$, respectively, a general message passing scheme can be described as follows:

$$X^{l+1} = \tau \odot F_W(A, X^l) + (1 - \tau) \odot X^l \quad (3)$$

where $\tau \in \mathbb{R}^{n \times c}$ is a multi-rate coefficient controlling the update rate of each each node (total n nodes) *w.r.t.* each feature (total c features). $F_W(A, X^l)$ is any graph coupled function involved with adjacency matrix A and current feature X^l with $W \in \mathbb{R}^{c \times c}$ denoting learnable weight parameters. For example, for GCN [3], the $F_W(\cdot)$ function can be summarized as $F_W(A, X^l) = AX^l$.

1) *Feature Updating Rate (τ):* G2-gating proposed by [11] alleviates oversmoothing through computing coefficient τ by graph gradient (which is essentially Dirichlet energy for a local node). Its intuitive idea is to forecast next step Dirichlet

energy before updating and stop the new updates if the forecast Dirichlet energy diminished.

On the one hand, the method is effective on overcoming oversmoothing problem. Directly connecting update rates to the Dirichlet energy helps prevent nodes from falling into aggregation of neighbors that are close to neighbors. In the meantime, G2-gating additionally gains node expressive power by extra-hop information hidden in the Dirichlet energy.

Leveraging Dirichlet energy to control feature updating rate, on the other hand, is limited by its non-parametric form and merely capturing the difference but not other relationships. In general, for each node v_i , its τ values with respect to each of its neighbors τ_{ij} ($v_j \in \Delta_{v_i}$) is controlled by the fixed mapping defined in Eq. (4).

$$\tau_{ij} = \tanh(\varepsilon_{DE}(\tilde{x}_i^l)) \quad (4)$$

The energy term in Eq. (4) is calculated using Eq. (5), where $p \in \mathbb{R}$ is a hyper-parameter.

$$\varepsilon_{DE}(x_i^l) = \sum_{x_b \in \Delta_{v_i}^x} |x_i^l - x_b^l|^p \quad (5)$$

An essential drawback of the coefficient controlling mechanism defined in Eqs. (4) and (5) is that the mapping is monotonically increasing with respect to the node difference. This hinders the method from modeling complex situations beyond node difference can modeling. For example, the updating rate τ_{ij} may be subject to a higher order function with respect to the sum between node v_i and its neighbors, which cannot be captured by Dirichlet energy. In addition, without having learnable parameters involved, a node cannot effectively determine and express its preference from supervised signals.

B. Graph Rhythm

As we mentioned earlier, local Dirichlet Energy is too strict and biased to account for all local nodes relation. The updating rate τ should depend more than simply the difference between central nodes and their neighbors but a generalized

and adaptive version. Alternatively, we propose a graph rhythm measure, defined in Eq. (6), to replace local Dirichlet Energy.

$$\varepsilon_{\text{GR}}(x_i^l) = \sum_{x_b \in \Delta_{v_i}^x} |(x_i^l || x_b^l) \Theta|^p \quad (6)$$

where $\Theta \in \mathbb{R}^{2c \times c}$ denote learnable parameters and $p \in \mathbb{R}$ is a hyper-parameter. We call Eq. (6) Graph Rhythm, which represents a generalized measure capable of capturing complex relationships between a central node v_i and its neighbors v_b to adaptively control feature updating rate.

Denote the feature aggregation function $F_W(A, X^l)$, one layer updating using graph rhythm based graph neural networks can be summarized as follows:

$$X^{l+1} = \tau \odot F_W(A, X^l) + (1 - \tau) \odot X^l \quad (7)$$

$$\tilde{X} = F_W(A, X^l) \quad (8)$$

$$\tau_{ij} = \tanh\left(\frac{1}{d_i} \sum_{x_b \in \Delta_{v_i}^x} |(\tilde{x}_i^l || \tilde{x}_b^l) \Theta|^p\right) \quad (9)$$

where $d_i = |\Delta_{v_i}|$ denotes node degree of v_i , $||$ is the concatenation operation, Θ and W are learnable weight parameters.

Advantages: We note following four main advantages of the graph rhythm:

- **Generalizable and Shareable:** Our measure can capture complex relationships in addition to neighbor difference, i.e, it adapts to a required relationship to τ driven by data instead of a static node difference relationship. Meanwhile, we maintain a shareable measure for all nodes to allow scalability.
- **Adaptable:** To guide the learning of measure mapped to τ , learnable parameters are involved in the measure computation to obtain feedback from supervised signal.
- **Locality:** Empirically, we observed performance degradation without explicit accounting for spatial information.
- **Permutation Invariant:** To hold the general invariant assumption for graph, our measure apply an aggregation function to preserve permutation invariant to local neighbors.

C. Graph Rhythm Omni-directional Relationship Mapping

We analyze the difference between local Dirichlet energy vs. the proposed graph rhythm in terms of their relationship mapping capabilities. To prove that graph rhythm is more generalizable in capturing complex node-neighbor relationship, we compare the following two terms and show that Eq. (4) (local Dirichlet Energy) can only capture node difference relationship whereas Eq. (11) (graph rhythm) can leverage learnable Θ parameters to capture complex relationships. For simplicity, we use $p = 1$ in the analysis. It is easily to show that same conclusion holds for $p > 1$:

$$\tau_{ij} = \tanh\left(\frac{1}{d_i} \sum_{x_b \in \Delta_{v_i}^x} |\tilde{x}_{ij}^l - \tilde{x}_{bj}^l|\right) \quad (10)$$

$$\tau_{ij} = \tanh\left(\frac{1}{d_i} \sum_{x_b \in \Delta_{v_i}^x} \sum_{k=1}^c |\Theta_{jk} \tilde{x}_{ik}^l + \Theta_{j(2k)} \tilde{x}_{bk}^l|\right) \quad (11)$$

Eq. (10) shows the mapping from Dirichlet Energy to τ , and Eq. (11) denotes the computation from graph rhythm to τ . In Eq. (11), if $\Theta_{jk} = 1$ for $k = j$ and 0 otherwise and $\Theta_{j(2k)} = -1$ for $2k = j$ and 0 otherwise, graph rhythm is reduced to the Dirichlet energy form, implying that graph rhythm can learn node difference relationship. Additionally, if Θ_{jk} and $\Theta_{j(2k)}$ have the same sign (both positive or negative), Eq. (11) can capture summation patterns, which cannot be captured by Dirichlet energy. In addition, graph rhythm determines a scalar feature updating rate by taking all features into consideration, which can help utilize cross-feature correlation to determine τ values.

D. GRN: Graph Rhythm Network

Since our proposed component is flexible to most standard residual-based MPNN networks, we use a standard MPNN setup in our experiment, followed by [11], for a consistent comparison. The framework is displayed in Figure 1. In the training stage, first, the feature space is masked with dropout rate and encoded with linear layer, then, we stack deep layers of our proposed GRN layers. In the end, we decode the hidden feature space to class labels and has a dropout rate for better regularization.

IV. EXPERIMENT

To examine the effectiveness of our proposed measure Graph Rhythm, we apply the method to both homophilic and heterophilic graphs and compare the performance with the existing SOTA baselines. We use the same dataset and splits from the previous baselines and therefore we report the baseline results directly obtained from original papers.

For the homophilic graph, we report the best results for each layer number in the range [2, 4, 8, 16, 32, 64, 128]. Each result is randomly searched within a range of hyperparameter settings and we report the best results. We choose the commonly used Cora dataset for our homophilic settings to verify the effective of GRN compared with G2-gating which leverages Dirichlet energy and dropout-based methods. For the aggregation function, we consistently apply Graph convolution layer as the fixed neighbor aggregation. The statistics of Cora dataset is shown in Table I (first row).

For the heterophilic graph, we report the best results for fixed 10 splits for dataset Texas, Wisconsin, Film, Squirrel, Chameleon, and Cornell [15], which covers small and middle level graphs. Their statistics are reported in Table I.

TABLE I
A SUMMARY OF THE BENCHMARK DATASET STATISTICS.

Datasets	Homophilic level	# of Nodes	# of Edges	# of Classes
Cora	0.81	2708	5429	7
Texas	0.11	183	295	5
Wisconsin	0.21	251	466	5
Film	0.22	7600	26752	5
Squirrel	0.22	5201	198493	5
Chameleon	0.23	2277	31421	5
Cornell	0.3	183	280	5

TABLE II
RESULTS OF GRAPH RHYTHM NETWORK (GRN) ON HETEROPHILIC GRAPH COMPARED TO EXISTING BASELINES ON 10-SPLIT DATA WITH MEAN AND STANDARD DEVIATION. FOR EACH DATASET, RED-COLORED TEXT DENOTES BEST RESULTS AND BLUE-COLORED TEXT DENOTES SECOND BEST RESULTS.

	Texas	Wisconsin	Film	Squirrel	Chameleon	Cornell
GGCN [16]	84.86 ± 4.55	86.86 ± 3.29	37.54 ± 1.56	55.17 ± 1.58	71.14 ± 1.84	85.68 ± 6.63
GPRGNN [17]	78.38 ± 4.36	82.94 ± 4.21	34.63 ± 1.22	31.61 ± 1.24	46.58 ± 1.71	80.27 ± 8.11
H2GCN [18]	84.86 ± 7.23	87.65 ± 4.98	35.70 ± 1.00	36.48 ± 1.86	60.11 ± 2.15	82.70 ± 5.28
FAGCN [19]	82.43 ± 6.89	82.94 ± 7.95	35.94 ± 1.78	42.59 ± 0.79	55.22 ± 3.19	79.19 ± 7.99
F2GAT [20]	82.70 ± 5.95	87.06 ± 4.13	36.65 ± 1.13	47.32 ± 2.43	67.81 ± 2.05	83.51 ± 6.70
MixHop [21]	77.84 ± 7.73	75.88 ± 4.90	32.22 ± 2.34	43.80 ± 1.48	60.50 ± 2.53	73.51 ± 6.34
GCNII [8]	77.57 ± 8.30	81.57 ± 4.30	37.44 ± 1.30	38.47 ± 1.58	63.86 ± 3.04	77.86 ± 3.79
Geom-GCN [15]	66.76 ± 2.72	64.51 ± 6.36	31.59 ± 1.15	38.15 ± 0.92	60.00 ± 2.81	60.54 ± 3.67
PairNorm [22]	60.27 ± 4.34	48.43 ± 6.14	27.40 ± 1.12	50.44 ± 2.04	62.74 ± 2.82	58.92 ± 3.15
LINKX [23]	74.60 ± 8.37	75.49 ± 4.52	32.14 ± 1.55	52.28 ± 1.55	65.22 ± 1.38	77.84 ± 8.51
GtoGNN [24]	84.32 ± 5.15	87.06 ± 5.53	37.35 ± 1.30	57.54 ± 1.39	69.78 ± 2.42	83.51 ± 4.26
GraphSAGE [5]	82.43 ± 6.14	81.18 ± 6.54	34.23 ± 0.99	41.61 ± 0.74	58.73 ± 1.68	75.95 ± 5.01
ResGatedGCN [25]	80.00 ± 5.57	81.57 ± 5.35	35.94 ± 1.13	37.60 ± 1.80	49.82 ± 2.71	73.51 ± 4.95
GCN [3]	55.14 ± 12.53	51.76 ± 6.36	27.31 ± 1.12	28.91 ± 1.10	55.22 ± 2.80	60.54 ± 5.30
GAT [4]	76.22 ± 11.19	69.41 ± 11.10	27.44 ± 0.89	36.77 ± 1.96	38.16 ± 1.58	61.89 ± 8.05
MLP	81.08 ± 4.75	85.29 ± 3.31	36.53 ± 0.70	28.77 ± 1.56	46.21 ± 2.99	81.89 ± 6.40
G2-GraphSAGE [11]	87.57 ± 3.86	87.84 ± 3.49	37.14 ± 1.01	64.26 ± 2.38	71.40 ± 2.38	86.22 ± 4.90
GRN: Graph Rhythm Network	89.73 ± 2.13	88.4 ± 3.36	37.2 ± 0.62	64.92 ± 2.32	71.73 ± 1.15	86.22 ± 2.0

TABLE III
AN EXAMPLE OF OUR LEARNED GRAPH RHYTHM MEASURE ON CORA DATASET FOR A 4 HIDDEN DIMENSION EMBEDDING, WE SHOW TWO OF THE FEATURES WITH MIXED PATTERN. CENTER IS THE TARGET NODE WEIGHT AND NEIGHBOR IS THE NEIGHBOR NODE WEIGHT.

	Feature 1			
center	-1.69E-02	-1.38E-03	2.75E-02	-5.04E-04
neighbor	-6.57E-03	-2.11E-02	2.62E-02	-2.13E-02
	Feature 2			
center	-1.12E-02	4.44E-02	1.05E-02	-5.51E-02
neighbor	2.04E+00	2.36E-03	3.29E-03	9.82E-04

TABLE IV
AVERAGE NUMBER OF LAYERS AND POWER P FOR EACH HETEROPHILIC DATASET OVER 10 SPLITS

Dataset	Texas	Wisconsin	Chameleon	Film	Squirrel	Cornell
Layers (l)	9	12.5	16	20	27	31
p	3.68	3.47	3.37	2.54	2.65	3.31

A. Dataset & Experiment Setup

Following [11], we test how our proposed model is affected by the increasing number of layers on a fixed split Cora dataset, and compare its performance with plain GCN, GCN with DropEdge, G2-gating that successfully leverages Dirichlet energy as two deep-architecture baselines. Following [11] and [16], we test how our model behaves on the heterophilic setting and select the same six heterophilic graphs first proposed by [15]. All six datasets have 10 fixed splits that have already been reported with many baselines. For each experiment, we use a 20-trials random hyperparameter search with the following range:

- *hidden dimension size*: [32, 64, 128, 256, 512]
- *Dropout & Dropin*: [0, 1]
- *layer number*: [5, 50]
- *learning rate*: [1E-3, 1E-2]
- *weight decay*: [1E-8, 1E-2]
- *power coefficient p*: [1, 5]

For the graph-coupled function $F_\theta(\cdot)$, we choose GraphSage aggregation for the heterophilic graphs, as they show best

performance gain in most cases. For Cora dataset, we use GCN aggregation, same as the G2-gating paper for consistent comparison. All experiments are run under NVIDIA V100 card with 32 CPU and 4 GPUs.

B. Results & Analysis

1) *Layer-wise Energy Preserving and Accuracy*: From Fig 2, it is observed that GRN preserves global Dirichlet energy through layers even though it doesn't control Dirichlet energy directly like G2-gating, confirming that our adaptive and generalized measure works in alleviating oversmoothing.

Fig 3 shows that GRN successfully alleviates oversmoothing and doesn't fall into uniform embeddings even after 128 layers, similarly for G2-GCN, noticing that such alleviation doesn't rely on Dirichlet energy measure, which answers our question that Dirichlet energy is not necessary condition to alleviate oversmoothing. In contrast, Dropout-GCN has degraded performance significantly after 32 layers. Meanwhile, GRN shows a clear performance increasing trend through layer increase, with its peak performance at 64 layers and outperforms all other cases, suggesting the effectiveness of GRN gaining expressive power with deep layers.

2) *Overall Performance*: Table II shows the results of GRN compared with existing baselines (reported by original authors on same dataset and split). The red color indicates best performance and blue color indicates second best performance, our results shows better performance on Texas, Wisconsin, Squirrel, and Chameleon dataset and is on par for Film and Cornell dataset. Meanwhile, we observe that for each split our methods converge to several fixed accuracy and has a lower variance in general, suggesting the robustness of GRN possibly brought by the learned measure that holds across splits.

3) *Omni-directional Relationship Mapping*: Table III shows one learned measure on Cora dataset with GRN. We can observe that in addition to a difference pattern between the target and its neighbors similar to Dirichlet energy, our learned measure also contains a summation pattern that provides a different updating direction. This empirically shows our proposed measure can capture complex node-neighbor relationships.

4) *Parameters*: Table IV reports the average number of layers and average power p for each dataset over 10 splits. The power p is observed to be stable around 3, suggesting a common measure complexity for these datasets. The average number of layers increase with the homophilic level increases. One possible reason for this phenomenon is the complexity of graph increases with the homophilic level approaches to 0.5.

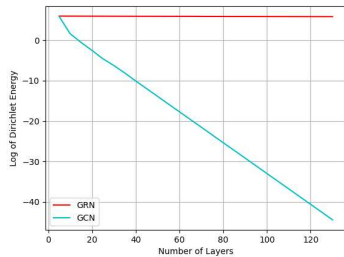


Fig. 2. Layer-wise Dirichlet energy for GRN and GCN. We use GCN as a baseline comparison to show that GRN preserves the Dirichlet energy even after 128 layers, showing that our proposed measure can control global Dirichlet energy without leveraging local Dirichlet energy.

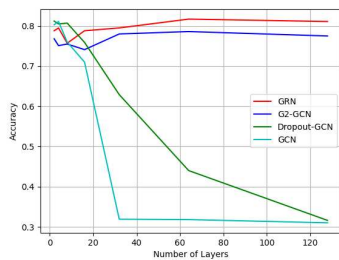


Fig. 3. Different model performance with the increasing number of layers for Cora dataset. We apply 10 random hyperparameter search trials for each choice of layers in the range [2,4,6,8,16,32,64,128] and report the best results for each method. GRN is our proposed method with GCN as backbone; G2-GCN is the existing STOAs framework with GCN as aggregation; Dropout-GCN is the GCN model with Dropout to alleviate oversmoothing. GCN is simply graph convolution networks.

V. CONCLUSION

In this paper, we propose to study how to learn very deep graph neural networks without compromising the downstream task performance. We argue that existing Dirichlet energy based approaches are only limited to characterize node difference, and the key to deliver informative deep graph neural networks is to properly model complex relationships between neighboring nodes to adaptively customize feature updating for each node. We propose graph rhythm, a new measure capable of learning omni-directional node relationships beyond Dirichlet energy can model. By integrating graph rhythm with residual message passing, a graph rhythm network (GRN) is proposed and shows superb performance as layer increases. Empirically, GRN is either better or on par with the existing STOAs on both homophilic and heterophilic networks.

ACKNOWLEDGEMENTS

This study is supported by the U.S. National Science Foundation under grant Nos. IIS-2236579, IIS-2302786 and IOS-2430224.

REFERENCES

- [1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. on Neural Networks and Learning Sys.*, vol. 32, no. 1, pp. 4–24, 2021.
- [2] Y. Jin, R. Gao, Y. He, and X. Zhu, "Gldl: Graph label distribution learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 12965–12974, Mar. 2024.
- [3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Intl. Conf. on Learning Rep.*, 2017.
- [4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.
- [5] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. of the 31st NeurIPS Conference*, NIPS'17, p. 1025–1035, 2017.
- [6] Y. Jin and X. Zhu, "ATNPA: A unified view of oversmoothing alleviation in graph neural networks," *ArXiv*, 2024.
- [7] T. Fang, Z. Xiao, C. Wang, J. Xu, X. Yang, and Y. Yang, "Dropmessage: Unifying random dropping for graph neural networks," *Proc. of the AAAI Conference*, vol. 37, p. 4267–4275, June 2023.
- [8] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proc. of ICML*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 1725–1735, 13–18 Jul 2020.
- [9] T. K. Rusch, M. M. Bronstein, and S. Mishra, "A survey on oversmoothing in graph neural networks," *arXiv:2303.10993*, 2023.
- [10] K. Zhou, X. Huang, D. Zha, R. Chen, L. Li, S.-H. Choi, and X. Hu, "Dirichlet energy constrained learning for deep graph neural networks," *Advances in neural information processing systems*, 2021.
- [11] T. K. Rusch, B. P. Chamberlain, M. W. Mahoney, *et al.*, "Gradient gating for deep multi-rate learning on graphs," in *ICLR*, 2023.
- [12] T. K. Rusch, B. P. Chamberlain, J. R. Rowbottom, S. Mishra, and M. M. Bronstein, "Graph-coupled oscillator networks," in *ICML*, 2022.
- [13] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *ICLR*, 2020.
- [14] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *ICLR*, 2020.
- [15] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," in *ICLR*, 2020.
- [16] Y. Yan, M. Hashemi, K. Swersky, Y. Yang, and D. Koutra, "Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks," in *ICDM*, pp. 1287–1292, dec 2022.
- [17] E. Chien, J. Peng, P. Li, and O. Milenkovic, "Adaptive universal generalized pagerank graph neural network," in *ICLR*, 2021.
- [18] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," in *NeurIPS*, vol. 33, pp. 7793–7804, 2020.
- [19] D. Bo, X. Wang, C. Shi, and H. Shen, "Beyond low-frequency information in graph convolutional networks," in *35th AAAI Conference on Artificial Intelligence, 2021*, pp. 3950–3957, 2021.
- [20] L. Wei, H. Zhao, and Z. He, "Designing the topology of graph neural networks: A novel feature fusion perspective," in *Proc. of the ACM Web Conference 2022*, WWW '22, ACM, Apr. 2022.
- [21] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan, "MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *Proc. of the ICML*, vol. 97, pp. 21–29, 09–15 Jun 2019.
- [22] L. Zhao and L. Akoglu, "Pairnorm: Tackling oversmoothing in gnns," in *ICLR*, 2020.
- [23] D. Lim, F. Hohne, X. Li, S. L. Huang, V. Gupta, O. Bhalerao, and S.-N. Lim, "Large scale learning on non-homophilous graphs: new benchmarks and strong simple methods," in *Proc. of the 35th Intl. Conf. on Neural Info. Processing Systems*, NIPS '21, 2024.
- [24] X. Li, R. Zhu, Y. Cheng, C. Shan, S. Luo, D. Li, and W. Qian, "Finding global homophily in graph neural networks when meeting heterophily," in *Proc. of the ICML*, vol. 162, pp. 13242–13256, 17–23 Jul 2022.
- [25] X. Bresson and T. Laurent, "Residual gated graph convnets," *ArXiv*, vol. abs/1711.07553, 2017.