GREAD: Graph Neural Reaction-Diffusion Networks

Jeongwhan Choi[1] **Seoyoung Hong**[1] **Noseong Park**[1] **Sung-Bae Cho**[1]

Abstract

Graph neural networks (GNNs) are one of the most popular research topics for deep learning. GNN methods typically have been designed on top of the graph signal processing theory. In particular, diffusion equations have been widely used for designing the core processing layer of GNNs, and therefore they are inevitably vulnerable to the notorious oversmoothing problem. Recently, a couple of papers paid attention to reaction equations in conjunctions with diffusion equations. However, they all consider limited forms of reaction equations. To this end, we present a reaction-diffusion equation-based GNN method that considers all popular types of reaction equations in addition to one special reaction equation designed by us. To our knowledge, our paper is one of the most comprehensive studies on reaction-diffusion equation-based GNNs. In our experiments with 9 datasets and 28 baselines, our method, called GREAD, outperforms them in a majority of cases. Further synthetic data experiments show that it mitigates the oversmoothing problem and works well for various homophily rates.

1. Introduction

Graphs are a useful data format that occurs frequently in real-world applications, e.g., computer vision and graphics (Monti et al., 2017), molecular chemistry inference (Gilmer et al., 2017), recommender systems (Ying et al., 2018; Choi et al., 2023b), drug discovery (Gaudelet et al., 2021), traffic forecasting (Choi et al., 2022), and so forth. With the rise of graph-based data, graph neural networks (GNNs) are attracting much attention these days. However, there have been fierce debates on the neural network architecture of GNNs (Kipf & Welling, 2017; Veličković et al., 2018; Defferrard et al., 2016; Wu et al.,

Table 1. A comparison table of existing methods. '△' means that it corresponds to a specific type of reaction only.

Model	Diffusion	Reaction	Continuous-time
FA-GCN	O	△	X
GPR-GNN	O	△	X
ACM-GCN	O	△	X
CGNN	O	X	O
GRAND	O	X	O
BLEND	O	X	O
GREAD	O	O	O

(a) $t = 0$ (b) $t = 5$ (c) $t = 25$ (d) $t = 50$

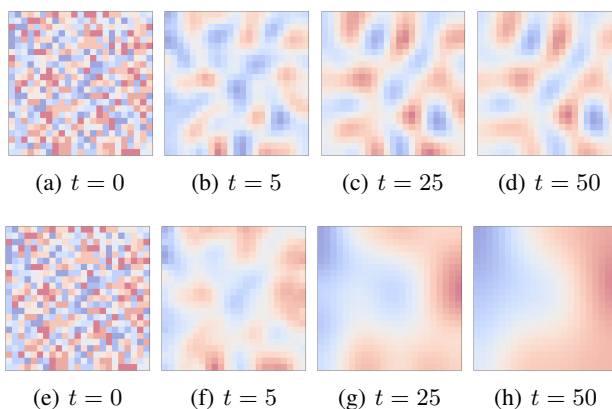(e) $t = 0$ (f) $t = 5$ (g) $t = 25$ (h) $t = 50$

Figure 1. An illustrative comparison between the diffusion equation in Eq. (11) (bottom) and our proposed blurring-sharpening (reaction-diffusion) equation in Eq. (14) (top) on a grid graph with one-dimensional node features. The diffusion equation causes the problem of oversmoothing while the reaction-diffusion seeks a balance between smoothing and sharpening.

2019; Chen et al., 2020; Chien et al., 2021; Zhu et al., 2020; Yan et al., 2022; Lim et al., 2021; Li et al., 2022; Luan et al., 2022; Rusch et al., 2022; Chamberlain et al., 2021b;a; Bodnar et al., 2022).

For the past couple of years, many proposed methods have been designed based on the diffusion concept. Many recent GNN methods that rely on low-pass filters fall into this category. Although they have shown non-trivial successes in many tasks, it is still unclear whether it is an optimal direction of designing GNNs.

In Table 1, we compare recent methods. Most of them

Table 2. The average ranking/accuracy and the Olympic ranking of some selected high-performing models on 9 real-world datasets. '∗' (resp. '†') indicates that an improvement over GloGNN (resp. ACM-GCN) is statistically significant ($p < 0.05$) under the Wilcoxon signed-rank test.

Method	Average		Olympic Ranking		
	Ranking	Accuracy	Gold	Silver	Bronze
GREAD-BS	1.56	76.64[∗†]	5	4	0
GREAD-FB∗	6.72	74.51[†]	0	0	3
GREAD-F	7.50	74.13	1	1	1
GloGNN	8.17	74.99	0	0	1
GREAD-AC	8.50	73.71	1	0	0
ACM-GCN	8.67	74.92	0	1	0
GGCN	9.50	75.05	0	1	0
Sheaf	10.33	75.06	1	1	0

rely on diffusion processes, while some of them (i.e., FA-GCN (Bo et al., 2021), GPR-GNN (Chien et al., 2021), and ACM-GCN (Luan et al., 2022)) partially utilize reaction processes. Those three methods, however, utilize limited forms of the reaction processes. In this regard, there do not exist any methods that fully consider diverse forms of reaction processes — we consider 7 reaction processes.

To this end, we propose the concept of graph neural reaction-diffusion network (GREAD), which is one of the most generalized architectures since we consider both the diffusion and the reaction processes. Reaction-diffusion equations are physical models that can be used when i) substances are diffused over space and time, and ii) they can sometimes react to each other. Whereas diffusion processes smooth node features on a graph out, reaction-diffusion processes lead to many local clusters that are also known as Turing patterns (Turing, 1952) (see Fig. 1). Since it is natural that nodes on a graph also constitute local clusters (in terms of class labels), we conjecture that reaction-diffusion equations are suitable for GNNs.

Our proposed model, GREAD, consists of three parts: an encoder, a reaction-diffusion layer, and an output layer (cf. Eqs. (6) to (8)). The reaction-diffusion layer has seven different types in its core part as shown in Eq. (10): i) Fisher (F), ii) Allen-Cahn (AC), iii) Zeldovich (Z), iv) Blurring-sharpening (BS), v) Source (S), vi) Filter Bank (FB), and vii) Filter Bank* (FB*). The first three reaction-diffusion equations are widely used in many natural science domains, e.g., biology, combustion, and so on, and the last three are used by some recent GNN methods (Xhonneux et al., 2020; Thorpe et al., 2022; Luan et al., 2022). In particular, the blurring-sharpening (BS) equation is designed by us and marks the best accuracy in many cases (cf. Table 2).

For our experiments, we consider 6 heterophilic and 3 homophilic datasets — heterophilic (resp. homophilic) means that neighboring nodes tend to have different (resp. similar)

classes. We also compare our method with a comprehensive set of 28 baselines, which covers early to recent GNNs. Our contributions can be summarized as follows:

1. We design a reaction-diffusion layer that incorporates seven types of reaction equations, including one type, called BS, proposed by us.
2. We carefully integrate the seven reaction equation types into our GNN method and customize its overall architecture for better accuracy. For instance, we use a soft adjacency matrix generating method, which shows a synergistic effect with the reaction-diffusion layer.
3. We consider a comprehensive set of 9 datasets and 28 baselines. Our method marks the best accuracy in many cases. The ranking and accuracy averaged over all the datasets are summarized in Table 2.

2. Preliminaries & Related Work

We first describe the meaning of the reaction-diffusion equation and various important GNN designs, followed by neural ordinary differential equations.

2.1. Reaction-Diffusion Equations

Reaction–diffusion equations are typically used to model the spatial and temporal change of the concentration of one or more chemical substances, i.e., substances are transformed into each other via local chemical reactions and spread out over a surface in space via diffusion. They are also frequently observed in other fields, such as biology, geology, physics (neutron diffusion theory), and ecology. In the field of graph machine learning, diffusion (resp. reaction) processes are typically carried out by applying low-pass (resp. high-pass) filters to graphs, which also corresponds to image blurring (resp. sharpening).

2.2. Graph Neural Networks

Notation Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a graph with node set $\mathcal{V}$ and edge set $\mathcal{E}$. The nodes are associated with a feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times F}$, where $|\mathcal{V}|$ denotes the number of nodes and F denotes the number of input features. $\mathbf{A}^{raw} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ is the adjacency matrix, where $\mathbf{A}^{raw}_{[i,j]}$ means the (i,j)-th element. The nodes are labelled by the index $i \in \mathcal{V}$, and one-hop neighborhood of each node is denoted as $\mathcal{N}_i$. The symmetric normalized Laplacian matrix, a commonly used feature aggregation matrix in GNNs, is defined as $\mathbf{L} := \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A}^{raw} \mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{A}$, where the diagonal degree matrix of $\mathbf{A}^{raw}$ is $\mathbf{D}$, and $\mathbf{A} := \mathbf{D}^{-1/2} \mathbf{A}^{raw} \mathbf{D}^{-1/2}$ is the symmetric normalized adjacency matrix — note that $\mathbf{A} \in [0,1]^{|\mathcal{V}| \times |\mathcal{V}|}$.

Graph Representation Learning GNNs (Kipf & Welling, 2017; Veličković et al., 2018; Hamilton et al., 2017;

Wu et al., 2019; Zhu & Koniusz, 2020) have many variants and applications. We focus on a brief introduction to the representation learning for nodes in supervised or semi-supervised classification tasks. Most existing approaches follow the message-passing framework constructed by stacking layers that propagate and aggregate node features.

The neighbor aggregation used in many existing GNNs implicitly exploits homogeneity and often fails to generalize to non-homogeneous graphs. Many existing GNNs operate as low-pass graph filters (Balcilar et al., 2021) that smooth features over the graph topology, which produces similar representations and as a result, similar predictions for neighboring nodes (Tiezzi et al., 2021; Oono & Suzuki, 2020; Li et al., 2018). Various GNNs were proposed to improve performance in low-homophily settings (Pei et al., 2020; Abu-El-Haija et al., 2019; Zhu et al., 2020; Chien et al., 2021; He et al., 2021; Lim et al., 2021; Luan et al., 2022; Bodnar et al., 2022; Di Giovanni et al., 2022; Li et al., 2022) and alleviate the oversmoothing problem. (Xu et al., 2018; Chen et al., 2020; Zhao & Akoglu, 2020; Rusch et al., 2022).

Diffusion on Graphs and Continuous GNNs The diffusion on graphs has recently been actively used in various applications (Freidlin & Wentzell, 1993; Freidlin & Sheu, 2000), including data clustering (Belkin & Niyogi, 2003; Coifman et al., 2005), image processing (Desquesnes et al., 2013; Elmoataz et al., 2008; Gilboa & Osher, 2009), and so on. It is to apply the following diffusion process to the feature matrix $\mathbf{X}$ of a graph:

$$\frac{d\mathbf{X}(t)}{dt} := div(\mathbf{A}(\mathbf{X}(t)))\nabla\mathbf{X}(t)) = -\mathbf{L}\mathbf{X}(t), \quad (1)$$

where div and ∇ are the divergence and the gradient operators, respectively. The initial features are evolved under the diffusion process to have the final representation. The diffusion equation and its unit-step Euler discretization can be defined as follows:

$$\mathbf{X}(t+1) = \mathbf{X}(t) - \mathbf{L}\mathbf{X}(t) = (\mathbf{I} - \mathbf{L})\mathbf{X}(t). \quad (2)$$

This is similar to GCN (Kipf & Welling, 2017) where the following augmented diffusion process with a weight matrix $\mathbf{W}$ and a nonlinear activation σ is used:

$$\sigma((\mathbf{I} - \mathbf{L})\mathbf{X}(t)\mathbf{W}). \quad (3)$$

From this perspective, several papers have proposed continuous-depth GNNs (Wang et al., 2021; Choi et al., 2021; Hwang et al., 2021; Thorpe et al., 2022; Choi et al., 2023a) inspired by the graph diffusion equation. One recent work is GRAND (Chamberlain et al., 2021b), which parameterizes the diffusion equation on graphs with a neural network. BLEND (Chamberlain et al., 2021a) used a

non-euclidean diffusion equation (known as Beltrami flow) to solve a joint positional feature space problem. These approaches contribute to non-trivial improvements in graph machine learning. We extend the diffusion to the reaction-diffusion equation in this work.

2.3. Neural Ordinary Differential Equations (NODEs)

Neural ordinary differential equations (NODEs) (Chen et al., 2018c) solve the initial value problem (IVP), which involves a Riemann integral problem, to calculate $\mathbf{h}(t_{i+1})$ from $\mathbf{h}(t_i)$:

$$\mathbf{h}(t_{i+1}) = \mathbf{h}(t_i) + \int_{t_i}^{t_{i+1}} f(\mathbf{h}(t_i), t; \theta_f)dt, \quad (4)$$

where the neural network parameterized by θ_f approximates the time-derivative of $\mathbf{h}$, i.e., $\dot{\mathbf{h}} \overset{\text{def}}{=} \frac{d\mathbf{h}(t)}{dt}$. We rely on various ODE solvers to solve the integral problem, from the explicit Euler method to the 4th order Runge–Kutta (RK4) method and the Dormand–Prince (DOPRI) method (Dormand & Prince, 1980). For instance, the Euler method is as follows:

$$\mathbf{h}(t + h) = \mathbf{h}(t) + \tau \cdot f(\mathbf{h}(t)), \quad (5)$$

where τ, which is usually smaller than 1, is a pre-configured step size. Eq. (5) is identical to a residual connection when $h = 1$ and therefore, NODEs are a continuous generalization of residual networks.

3. Proposed Method

After describing an overview of our method, we describe its detailed designs, followed by its training algorithm. The theoretical and empirical complexity analyses are in Appendix D and F.3, respectively.

3.1. Overview of GREAD

Given a graph $\mathcal{G}$ with its node feature matrix $\mathbf{X}$, its symmetric normalized Laplacian matrix $\mathbf{L}$, and its symmetric normalized adjacency $\mathbf{A}$, the overall architecture of GREAD can be written as follows — instead of $\mathbf{A}$, we can also use a generated soft adjacency matrix $\tilde{\mathbf{A}}$, which will be described in the next subsection:

$$\mathbf{H}(0) = \mathbf{e}(\mathbf{X}) \quad \text{(Encoding layer)}, \quad (6)$$

$$\mathbf{H}(T) = \mathbf{H}(0) + \int_0^T \mathbf{f}(\mathbf{H}(t))dt \quad \text{(Reac.-diff. layer)}, \quad (7)$$

$$\hat{\mathbf{y}} = \mathbf{o}(\mathbf{H}(T)) \quad \text{(Output layer)}, \quad (8)$$

where $\mathbf{f}(\mathbf{H}(t)) := \frac{d\mathbf{H}(t)}{dt} = -\alpha\mathbf{L}\mathbf{H}(t) + \beta\mathbf{r}(\mathbf{H}(t))$ is in the reaction-diffusion form. $\mathbf{r}(\mathbf{H})$ is a reaction term, and α and β are trainable parameters to (de-)emphasize each term. $\mathbf{e}$ is an encoder embeds the node feature matrix $\mathbf{X}$ into an

initial hidden state $\mathbf{H}(0)$. We then evolve the initial hidden state to $\mathbf{H}(T)$ via the reaction-diffusion equation of $\mathbf{f}$. The function $\mathbf{o}$ is an output layer for a downstream task, e.g., node classification. In particular, β can be either a scalar or a vector parameter, where the scalar setting means that we apply the same reaction process to all nodes and in the vector setting, we apply different reaction processes with different coefficients to nodes.

The encoder $\mathbf{e}$ has a couple of fully-connected layers with rectified linear unit (ReLU) activations. The output layer $\mathbf{o}$ is typically a fully-connected layer, followed by a softmax activation for classification in our experiments.

In particular, we consider almost all existing reaction terms for $\mathbf{r}$, which is different from existing works that do not consider them in a thorough manner. In this perspective, our work is the most comprehensive study on reaction-diffusion GNNs to our knowledge. In the following subsection, we also show that some choices of the reaction term correspond to other famous models — in other words, some other famous models are special cases of GREAD.

3.2. Soft Adjacency Matrix Generation

Given a graph $\mathcal{G}$, one can use its original symmetric normalized adjacency matrix $\mathbf{A} \in [0,1]^{|\mathcal{V}| \times |\mathcal{V}|}$ for $\mathbf{f}(\mathbf{H}(t))$. However, we also provide the method to generate a soft adjacency matrix, denoted $\tilde{\mathbf{A}} \in [0,1]^{|\mathcal{V}| \times |\mathcal{V}|}$ — we use $\tilde{\mathbf{L}}$ to denote the Laplacian counterpart of $\tilde{\mathbf{A}}$. Our soft adjacency matrix plays a crucial role in learning diffusivity. Our reaction-diffusion layer uses the soft adjacency matrix to learn the diffusivity.

In order to generate such soft adjacency matrices, we use the scaled dot product method (Vaswani et al., 2017):

$$\tilde{\mathbf{A}}_{[i,j]} := softmax\left(\frac{(\mathbf{W}_K \mathbf{H}_i)^T \mathbf{W}_Q \mathbf{H}_j}{d_K}\right), \quad (9)$$

where $\tilde{\mathbf{A}}_{[i,j]}$ means the (i,j)-th element of $\tilde{\mathbf{A}}$, $\mathbf{W}_K$ and $\mathbf{W}_Q$ are trainable parameters, and d_K is the scale factor. $\mathbf{H}_i, \mathbf{H}_j$ are trainable embedding vectors of nodes i, j.

3.3. Reaction-diffusion Layer

Eq. (7) is our method's main processing layer, called the reaction-diffusion layer. Given the definition of $\mathbf{f}$, '$-\mathbf{LH}(t)$' is a diffusion term that corresponds to the heat equation describing the spread of heat over $\mathcal{G}$ and has been used widely by various GNNs (Wang et al., 2021; Choi et al., 2021; Chamberlain et al., 2021b). It is known that the diffusion term causes the problem of oversmoothing, which means that the last hidden states of nodes become too similar when applying only the diffusion processing too much. To this end, many models prefer shallow architectures that do not cause the oversmoothing problem (Wu et al., 2019;

Kipf & Welling, 2017) or use heuristic methods to prevent it (Zhao & Akoglu, 2020; Chen et al., 2018a; 2020; Li et al., 2019; Liu et al., 2020; Huang et al., 2018; Chen et al., 2018b).

In our case, we prevent the oversmoothing problem by adding the reaction term $\mathbf{r}$ and solving Eq. (7) with ODE solvers (Dormand & Prince, 1980). In other words, our reaction-diffusion layer is continuous, which is yet another distinguishing point of our method since many other models are based on discrete layers (Kipf & Welling, 2017; Bo et al., 2021; Chien et al., 2021; Zhu et al., 2020; Hamilton et al., 2017). We consider the following options for $\mathbf{r}$:

$$\mathbf{r}(\mathbf{H}(t)) := \begin{cases} \mathbf{H}(t) \odot (1 - \mathbf{H}(t)), & \text{if Fisher (F)} \\ \mathbf{H}(t) \odot (1 - \mathbf{H}(t)^{\circ 2}), & \text{if Allen-Cahn (AC)} \\ \mathbf{H}(t) \odot (\mathbf{H}(t) - \mathbf{H}(t)^{\circ 2}), & \text{if Zeldovich (Z)} \\ (\tilde{\mathbf{A}} - \tilde{\mathbf{A}}^2)\mathbf{H}(t), & \text{if Blurring-Sharpening (BS)} \\ \mathbf{H}(0), & \text{if Source Term (ST)} \\ \mathbf{LH}(t), & \text{if Filter Bank (FB)} \\ \mathbf{LH}(t) + \mathbf{H}(t), & \text{if Filter Bank* (FB*)} \end{cases}$$

(10)

where '$\odot$' means the Hadamard product, and '$\circ 2$' means the Hadamard power.

The first three reaction terms, i.e., F, AC, and Z, are widely used in various domains. For instance, F is used to describe the spreading of biological populations (Fisher, 1937), and AC is used for describing the phase separation process in multi-component alloy systems, which includes order-disorder transitions (Allen & Cahn, 1979). Z is a generalized equation that describes the phenomena that occur in combustion theory (Gilding & Kersner, 2004). The last BS is specially designed by us for GNNs, which we will describe shortly. ST is a case where the initial hidden state is added as a reaction term (Xhonneux et al., 2020). ST is not theoretically a reaction process, but we consider it as part of our model since their goals are the same, i.e., alleviating the notorious oversmoothing problem. FB means high-pass filters that correspond to reaction processes. By adding a high-pass filter, our reaction-diffusion layer acts like a filter bank holding the low and high-pass filters. FB* is a reaction term that also considers the identity channel $\mathbf{H}(t)$.

Blurring-Sharpening (BS) Given the reaction-diffusion layer in Eq. (7), the proposed blurring-sharpening (BS) process, whose time-derivative of $\mathbf{H}(t)$ will be defined in Eq. (14), is to perform the blurring (diffusion) and the sharpening (reaction) operations alternately in the layer. We show that our proposed blurring-sharpening process reduces to a certain form of the reaction-diffusion process. Many GNNs can be generalized to the following blurring (or diffusion) process, i.e., the low-pass graph convolutional filtering for

Algorithm 1 How to train our proposed GREAD

Input: Training data D_{train}, Validating data D_{val}, Maximum iteration number max_iter
Initialize model parameters θ;
$k \leftarrow 0$;
while $k < max\_iter$ **do**
     Construct a mini-batch B from D_{train}
     Train θ with Eq. (15) and B;
     Validate and update the best parameters θ^* with D_{val}
     $k \leftarrow k + 1$;
end
return θ^*;

---

blurring. We also use the same blurring operation at first:

$$\begin{aligned}\mathbf{B}(t+h) &= \mathbf{H}(t) - \tilde{\mathbf{L}}\mathbf{H}(t), \\ &\Rightarrow \mathbf{H}(t) + (\tilde{\mathbf{A}} - \mathbf{I})\mathbf{H}(t), \qquad (11) \\ &\Rightarrow \tilde{\mathbf{A}}\mathbf{H}(t).\end{aligned}$$

We then propose to apply the following high-pass graph convolutional filtering or sharpening process to $\mathbf{B}(t + h)$. In other words, there is a sharpening process following the above blurring process in a layer as follows — the full derivation is in Appendix C:

$$\begin{aligned}\mathbf{H}(t+h) &= \mathbf{B}(t+h) + \tilde{\mathbf{L}}(\mathbf{B}(t+h)), \\ &\Rightarrow \mathbf{H}(t) - \tilde{\mathbf{L}}\mathbf{H}(t) + (\tilde{\mathbf{A}} - \tilde{\mathbf{A}}^2)\mathbf{H}(t).\end{aligned} \qquad (12)$$

Therefore, we can derive the following difference equation:

$$\mathbf{H}(t+h) - \mathbf{H}(t) = -\tilde{\mathbf{L}}\mathbf{H}(t) + (\tilde{\mathbf{A}} - \tilde{\mathbf{A}}^2)\mathbf{H}(t). \quad (13)$$

After taking the limit of $h \to 0$ and adding the coefficients α, β,

$$\mathbf{f}(\mathbf{H}(t)) := \frac{d\mathbf{H}(t)}{dt} = -\alpha\tilde{\mathbf{L}}\mathbf{H}(t) + \beta(\tilde{\mathbf{A}} - \tilde{\mathbf{A}}^2)\mathbf{H}(t), \quad (14)$$

which is a reaction-diffusion equation where $\mathbf{r}(\mathbf{H}(t)) := (\tilde{\mathbf{A}} - \tilde{\mathbf{A}}^2)\mathbf{H}(t)$. Therefore, our proposed method, BS, uses the reaction-diffusion layer in Eq. (7) with the specific time-derivative definition of Eq. (14).

3.4. Training Algorithm

We use Alg. (1) to train our proposed model. The full training process minimizes the cross-entropy loss:

$$\mathcal{L} := \sum_{i}^{n} \mathbf{y}_i^T \log \hat{\mathbf{y}}_i, \qquad (15)$$

where $\mathbf{y}_i$ is the one-hot ground truth vector of i-th training sample, and $\hat{\mathbf{y}}_i$ is its inference outcome by our model.

3.5. Comparison with GNNs

When ST is used, GREAD is analogous to GCNII in the perspective that both methods inject the initial hidden state. GREAD, FA-GCN, and GPR-GNN differ in how to utilize low and high-pass filters. FA-GCN learns edge-level aggregation weights as in GAT but allows negative weights. GPR-GNN uses learnable weights that can be both positive and negative for feature propagation. Those enable FA-GCN and GPR-GNN to adapt to heterophilic graphs and to handle both high and low-frequency parts of graph signals. However, GREAD-BS sharpens low-pass filtered signals following our developed reaction-diffusion system. GREAD-BS also adaptively adjusts each term.

We also compare with some continuous-time GNN models. CGNN can be derived from the reaction-diffusion layer in Eq. (7) with $\mathbf{L}$ by setting $\mathbf{f}$ with $\mathbf{r}(\mathbf{H}(t)) := \mathbf{H}(0)$ and using a weight parameter $\mathbf{W}$:

$$\mathbf{f}(\mathbf{H}(t))_{\text{CGNN}} := -\mathbf{L}\mathbf{H}(t) + \mathbf{H}(t)\mathbf{W} + \mathbf{H}(0). \quad (16)$$

The linear GRAND model corresponds to using only our diffusion process:

$$\mathbf{f}(\mathbf{H}(t))_{\text{GRAND}} := -\tilde{\mathbf{L}}\mathbf{H}(t) = -(\mathbf{I} - \tilde{\mathbf{A}})\mathbf{H}(t). \quad (17)$$

We note that two continuous models can not capture high frequency parts. In particular, GRAND does not use any reaction term.

4. Experiments

We first compare our method with other baselines for node classification tasks. We then discuss the ability of mitigating oversmoothing on a synthetic graph and show the experiment with different heterophily levels on other synthetic graphs. Our code is available at https://github.com/jeongwhanchoi/GREAD.

4.1. Node Classification on Real-world Datasets

Real-world Datasets We now evaluate the performance of GREAD and existing GNNs on a variety of real-world datasets. We consider 6 heterophilic datasets with low homophily ratios used in (Pei et al., 2020): i,ii) Chameleon, Squirrel (Rozemberczki et al., 2021), iii) Film (Tang et al., 2009), iv, v, vi) Texas, Wisconsin and Cornell from WebKB. We also test on 3 homophilic graphs with high homophily ratios: i) Cora (McCallum et al., 2000), ii) CiteSeer (Sen et al., 2008), iii) PubMed (Yang et al., 2016). Table 3 summarizes the number/size of nodes, edges, classes, features, and the homophily ratio. We use the dataset splits taken from (Pei et al., 2020). We report the mean and standard deviation accuracy after running each experiment with 10 fixed train/val/test splits.

Table 3. Benchmark dataset properties and statistics

Dataset	Texas	Wisconsin	Cornell	Film	Squirrel	Chameleon	Cora	Citeseer	PubMed
Classes	5	5	5	5	5	5	6	7	3
Features	1,703	1,703	1,703	932	2,089	235	1,433	3,703	500
Nodes	183	251	183	7,600	5,201	2,277	2,708	3,327	19,717
Edges	279	466	277	26,752	198,353	31,371	5,278	4,552	44,324
Hom. ratio	0.11	0.21	0.30	0.22	0.22	0.23	0.81	0.74	0.80

Table 4. Results on real-world datasets: mean $\pm$ std. dev. accuracy for 10 different data splits. We show the best three methods in red (first), blue (second), and purple (third). Other missing 16 baselines are in Appendix B.

| Dataset	Texas	Wisconsin	Cornell	Film	Squirrel	Chameleon	Cora	Citeseer	PubMed	Avg.
Geom-GCN	66.76±2.72	64.51±3.66	60.54±3.67	31.59±1.15	38.15±0.92	60.00±2.81	85.35±1.57	78.02±1.15	89.95±0.47	63.87
H2GCN	84.86±7.23	87.65±4.98	82.70±5.28	35.70±1.00	36.48±1.86	60.11±2.15	87.87±1.20	77.11±1.57	89.49±0.38	71.33
GGCN	84.86±4.55	86.86±3.29	85.68±6.63	37.54±1.56	55.17±1.58	71.14±1.84	87.95±1.05	77.14±1.45	89.15±0.37	75.05
LINKX	74.60±8.37	75.49±5.72	77.84±5.81	36.10±1.55	61.81±1.80	68.42±1.38	84.64±1.13	73.19±0.99	87.86±0.77	71.11
GloGNN	84.32±4.15	87.06±3.53	83.51±4.26	37.35±1.30	57.54±1.39	69.78±2.42	88.31±1.13	77.41±1.65	89.62±0.35	74.99
ACM-GCN	87.84±4.40	88.43±3.22	85.14±6.07	36.28±1.09	54.40±1.88	66.93±1.85	87.91±0.95	77.32±1.70	90.00±0.52	74.92
GCNII	77.57±3.83	80.39±3.40	77.86±3.79	37.44±1.30	38.47±1.58	63.86±3.04	88.37±1.25	77.33±1.48	90.15±0.43	70.16
CGNN	71.35±4.05	74.31±7.26	66.22±7.69	35.95±0.86	29.24±1.09	46.89±1.66	87.10±1.35	76.91±1.81	87.70±0.49	63.96
GRAND	75.68±7.25	79.41±3.64	82.16±7.09	35.62±1.01	40.05±1.50	54.67±2.54	87.36±0.96	76.46±1.77	89.02±0.51	68.94
BLEND	83.24±4.65	84.12±3.56	85.95±6.82	35.63±1.01	43.06±1.39	60.11±2.09	88.09±1.22	76.63±1.60	89.24±0.42	71.79
Sheaf	85.05±5.51	89.41±4.74	84.86±4.71	37.81±1.15	56.34±1.32	68.04±1.58	86.90±1.13	76.70±1.57	89.49±0.40	75.06
GRAFF	88.38±4.53	87.45±2.94	83.24±6.49	36.09±0.81	54.52±1.37	71.08±1.75	87.61±0.97	76.92±1.70	88.95±0.52	74.92
GREAD-BS	88.92±3.72	89.41±3.30	86.49±7.15	37.90±1.17	59.22±1.44	71.38±1.31	88.57±0.66	77.60±1.81	90.23±0.55	76.64
GREAD-F	89.73±4.49	86.47±4.84	86.49±5.13	36.72±0.66	46.16±1.44	65.20±1.40	88.39±0.91	77.40±1.54	90.09±0.31	74.13
GREAD-AC	85.95±2.65	86.08±3.56	87.03±4.95	37.21±1.10	45.10±2.11	65.09±1.08	88.29±0.67	77.38±1.53	90.10±0.36	73.71
GREAD-Z	87.30±5.68	86.29±4.32	85.68±5.41	37.01±1.11	46.25±1.72	62.70±2.30	88.31±1.10	77.39±1.90	90.11±0.27	73.45
GREAD-ST	81.08±5.67	86.67±3.01	86.22±5.98	37.66±0.90	45.83±1.40	63.03±1.32	88.47±1.19	77.25±1.47	90.13±0.36	72.93
GREAD-FB	86.76±5.05	87.65±3.17	86.22±5.85	37.40±0.55	50.83±2.27	66.05±1.21	88.03±0.78	77.28±1.73	90.07±0.45	74.48
GREAD-FB*	87.03±3.97	88.04±1.63	85.95±5.64	37.70±0.51	50.57±1.52	65.83±1.10	88.01±0.80	77.42±1.93	90.08±0.46	74.51

(a) $t = 0$ (b) $t = 1$ (c) $t = 3.5$
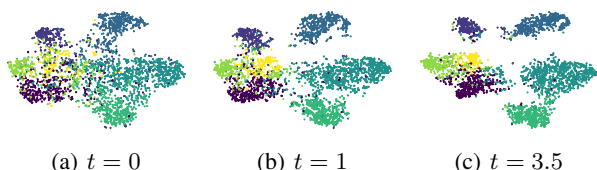
Figure 2. The snapshots of the evolution process of the node feature at various ODE time points in GREAD for Cora. Different colors correspond to different ground truth classes. More visualizations in other datasets are in Appendix F.4.

Baselines We use a comprehensive set of baselines classified into the following four groups:

1. In the first group of baselines, we consider classical GNN methods: ChebNet (Defferrard et al., 2016), GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), GraphSAGE (Hamilton et al., 2017), and SGC (Wu et al., 2019).

2. The next group includes the GNN methods designed for heterophilic settings: MixHop (Abu-El-Haija et al., 2019), Geom-GCN (Pei et al., 2020), H2GCN (Zhu et al., 2020), FA-GCN (Bo et al., 2021), GPR-GNN (Chien et al., 2021), WRGAT (Suresh et al., 2021), GGCN (Yan et al., 2022), LINKX (Lim et al., 2021), GloGNN (Li et al., 2022) and ACM-GCN (Luan et al., 2022).

3. The third group has GNN methods tackling the over-smoothing problem: PairNorm (Zhao & Akoglu, 2020), JKNet (Xu et al., 2018), GCNII (Chen et al., 2020), and GCON (Rusch et al., 2022).

4. The last group contains continuous-time GNN methods: GDE (Poli et al., 2019), CGNN (Xhonneux et al., 2020), GRAND (Chamberlain et al., 2021b), BLEND (Chamberlain et al., 2021a), ACMP (Wang et al., 2023), Sheaf (Bodnar et al., 2022), and GRAFF (Di Giovanni et al., 2022).

Hyperparameters For our method, we test with the following hyperparameter configurations: we train for 200 epochs using the Adam optimizer. The detailed search space and other hyperparameters are in Appendix E.2. We also list the best hyperparameter configuration for each data in Appendix E.2. If a baseline's accuracy is known and its

Table 5. Ablation study on soft adjacency matrix. More results in other datasets are in Appendix F.1.

Dataset	**A**	BS	F	AC	Z	ST	FB	FB*
Cornell	OA	85.14	85.41	83.51	83.78	85.41	85.19	83.90
	SA	**86.49**	**86.49**	**87.03**	**85.68**	**86.22**	**86.22**	**85.95**
Film	OA	37.24	36.68	35.93	36.04	37.43	37.18	37.13
	SA	**37.90**	**37.20**	**37.21**	**37.01**	**37.66**	**37.40**	**37.70**

Table 6. Ablation study on β. More results in other datasets are in Appendix F.1.

Dataset	**A**	BS	F	AC	Z	ST	FB	FB*
Texas	SC	81.35	84.05	85.41	**87.30**	79.73	76.95	77.08
	VC	**88.92**	**89.73**	85.95	86.49	**81.08**	**86.76**	**87.03**
Wisconsin	SC	84.71	85.69	**86.08**	84.51	83.52	84.24	86.21
	VC	**89.41**	**86.47**	85.69	**86.29**	**86.67**	**87.65**	**88.04**

experimental environments are the same as ours, we use the officially announced accuracy. If not, we execute a baseline using its official codes and the hyperparameter search procedures based on their suggested hyperparameter ranges.

Experimental Results   Table 2 shows the average ranking and accuracy in all the real-world datasets. GREAD-BS is ranked at the top with the average ranking of 1.56. GREAD-BS shows a clearly higher ranking in comparison with GloGNN and others. In Fig. 2, we visualize the hidden node features at each ODE time step of Eq. (7), and the reaction-diffusion processes of GREAD lead to local clusters after several steps.

Table 4 presents the detailed classification performance. As reported, our method marks the best accuracy in all cases except for Squirrel and Citeseer. GloGNN and Sheaf show comparable accuracy values from time to time. However, there are no existing methods that are as stable as GREAD-BS. For example, GCNII shows reasonably high accuracy in homophilic datasets, but not heterophilic ones. Sheaf shows the best or the second-best place in merely two cases. While GREAD-BS is the best method overall, GREAD-F is the best method for Texas and is the second-best for Cornell. GREAD-AC marks the best accuracy on Cornell.

Ablation Studies   We conduct ablation studies about the soft adjacency matrix generation. GREAD can use both the original symmetric normalized adjacency matrix, denoted as OA, and the soft adjacency matrix denoted as SA. We compare both options. As reported in Table 5, SA increases the model accuracy in Cornell and Film.

Next, we also perform the ablation study on β. β can be either a scalar parameter (denoted as SC) or a learnable vector parameter (denoted as VC). We compare them in
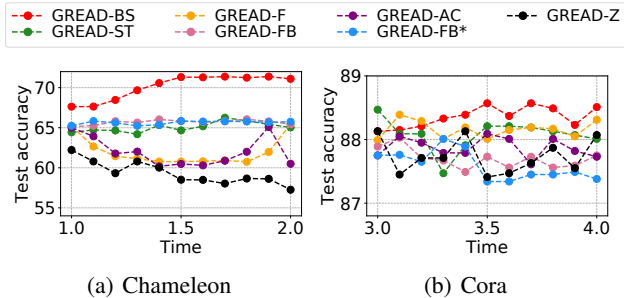

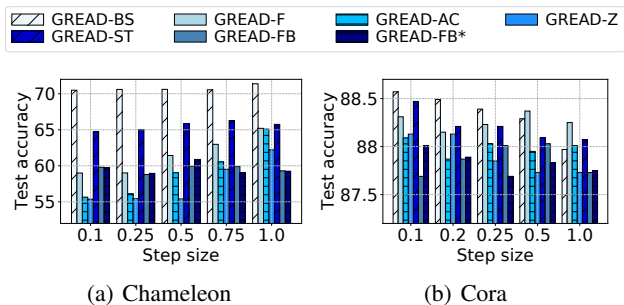*Figure 3.* Sensitivity to T. More results in other datasets are in Appendix F.2.

Figure 4. Sensitivity to the ODE step size τ. More results in other datasets are in Appendix F.2.

Table 6. VC shows effectiveness for most cases. The VC setting creates a reaction-diffusion process rich enough to classify nodes, as shown in Fig. 2.

Sensitivity w.r.t. the Terminal Integration Time T By varying T in Eq. (7), we investigate how the model accuracy changes. The detailed results are in Fig. 3. In Chameleon, GREAD-BS achieves the highest mean test accuracy at $T = 1.9$.

Sensitivity w.r.t. the ODE Step Size τ Fig. 4 shows the mean test accuracy by varying the step size τ of RK4. In Chameleon, GREAD-BS shows stable test accuracy at all the step sizes, whereas in Cora, GREAD-BS tends to show higher accuracy with larger step sizes.

4.2. Oversmoothing and Dirichlet Energy

The Dirichlet Energy   We can analyze the degree of over-smoothing (Nt & Maehara, 2019; Oono & Suzuki, 2020) from the perspective of the Dirichlet energy (Rusch et al., 2022; 2023). The Dirichlet energy $E(\mathbf{H}, \mathbf{A}^{raw})$ on the node hidden feature $\mathbf{H}$ of an undirected graph $\mathcal{G}$ is defined as follows:

$$E(\mathbf{H}, \mathbf{A}^{raw}) = \frac{1}{N} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} \mathbf{A}^{raw}_{[i,j]} ||\mathbf{H}_i - \mathbf{H}_j||^2, \quad (18)$$
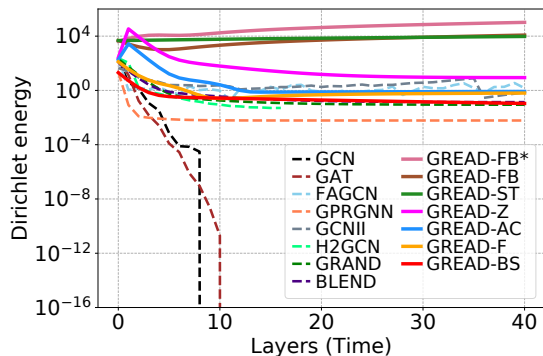
Figure 5. Evolution of the Dirichlet energy on the synthetic random graph. The Y-axis is the logarithmic Dirichlet energy in each layer's output given a GNN of 40 layers.

where $\mathbf{H}_i, \mathbf{H}_j$ mean i-th and j-th rows, respectively.

The oversmoothing phenomenon means that as the depth increases, all node features converge to constants. Thus, $E(\mathbf{H}, \mathbf{A})$ decays to zero asymptotically in time. We will show, via the evolution of the Dirichlet energy, that our proposed method mitigates the oversmoothing problem.

Experimental Environments We use the synthetic dataset, called cSBMs (Deshpande et al., 2018), to demonstrate the mitigation of oversmoothing. This synthetic data is an undirected graph representing 100 nodes in a two-dimensional space with two classes randomly connected with a probability of $p = 0.9$. We report the layer-wise Dirichlet energy given a GNN of 40 layers.

Experimental Results Fig. 5 demonstrates traditional GNNs, such as GCN, and GAT, suffer from oversmoothing because the Dirichlet energy decays exponentially to zero in the first five layers. Converging to zero indicates that the node features become constant, while GREAD has no such behaviors. The Dirichlet energy of GREAD can be bounded in time thanks to the reaction term. GRAND only has a diffusion term with learned diffusivity, so that it can delay the oversmoothing. In the case of H2GCN, it is impossible to report on deeper layers due to memory limitations.

4.3. Different Homophily Levels

Experimental Environments In order to test the classification capability of GNNs, we use the synthetic Cora generator (Zhu et al., 2020; Li et al., 2021). We generate synthetic graphs with various homophily ratios and report the test accuracy.

Experimental Results Fig. 6 shows the mean test accuracy on all random splits of the synthetic Cora datasets. MLP, which does not consider the connectivity of nodes,

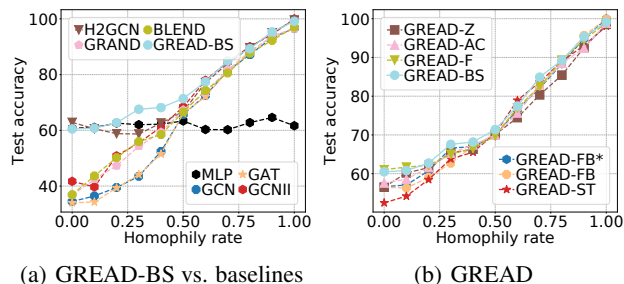(a) GREAD-BS vs. baselines   (b) GREAD

Figure 6. Experiments on the synthetic Cora with controlled homophily rates.

maintains its test accuracy for all homophily rates, which is obvious. GCN, GAT, and GRAND, which consider only diffusion, perform poorly at low homophily settings. H2GCN shows reasonable performance on low homophily rates, but its accuracy suddenly decreases at some homophily settings. All GREAD models have the best trend overall without sudden drops. The reaction terms of GREAD contribute to their stable accuracy for both homophily and heterophily settings compared with other models that rely on only diffusion processes, such as GCN and GRAND.

5. Conclusions

We presented the concept of graph neural reaction-diffusion equation, called GREAD. Our proposed GREAD is one of the most generalized architectures considering both the diffusion and reaction processes. We design a reaction-diffusion layer that has three types of reaction equations widely used in natural sciences. We also add four reaction terms, including one special reaction term called Blurring-sharpening (BS) designed by us for GNNs. Therefore, our reaction-diffusion layer has seven types. We consider a comprehensive set of 9 real-world datasets with various homophily difficulties and 28 baselines. GREAD marks the best accuracy in almost all cases. In our experiments with the two kinds of synthetic datasets, GREAD shows that it alleviates the oversmoothing problem and performs well on various homophily rates. This shows that our proposed model is a novel framework for constructing GNNs using the concept of the reaction-diffusion equation.

Acknowledgement

References

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Ver Steeg, G., and Galstyan, A. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*, pp. 21–29, 2019.

Allen, S. M. and Cahn, J. W. A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening. *Acta metallurgica*, 27(6):1085–1095, 1979.

Balcilar, M., Renton, G., Héroux, P., Gaüzère, B., Adam, S., and Honeine, P. Analyzing the expressive power of graph neural networks in a spectral perspective. In *ICLR*, 2021.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Biewald, L. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Bo, D., Wang, X., Shi, C., and Shen, H. Beyond low-frequency information in graph convolutional networks. In *AAAI*, 2021.

Bodnar, C., Giovanni, F. D., Chamberlain, B. P., Lio, P., and Bronstein, M. M. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In *NeurIPS*, 2022.

Chamberlain, B. P., Rowbottom, J., Eynard, D., Di Giovanni, F., Xiaowen, D., and Bronstein, M. M. Beltrami flow and neural diffusion on graphs. In *NeurIPS*, 2021a.

Chamberlain, B. P., Rowbottom, J., Goronova, M., Webb, S., Rossi, E., and Bronstein, M. M. GRAND: Graph neural diffusion. In *ICML*, 2021b.

Chen, J., Ma, T., and Xiao, C. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *ICLR*, 2018a.

Chen, J., Zhu, J., and Song, L. Stochastic training of graph convolutional networks with variance reduction. In *ICML*, 2018b.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *ICML*, 2020.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *NeurIPS*, 2018c.

Chien, E., Peng, J., Li, P., and Milenkovic, O. Adaptive universal generalized pagerank graph neural network. In *ICLR*, 2021.

Choi, H., Choi, J., Hwang, J., Lee, K., Lee, D., and Park, N. Climate modeling with neural advection–diffusion equation. *Knowledge and Information Systems*, pp. 1–25, 2023a.

Choi, J., Jeon, J., and Park, N. LT-OCF: Learnable-time ode-based collaborative filtering. In *CIKM*, 2021.

Choi, J., Choi, H., Hwang, J., and Park, N. Graph neural controlled differential equations for traffic forecasting. In *AAAI*, 2022.

Choi, J., Hong, S., Park, N., and Cho, S.-B. Blurring-sharpening process models for collaborative filtering. In *SIGIR*. ACM, 2023b.

Coifman, R. R., Lafon, S., Lee, A. B., Maggioni, M., Nadler, B., Warner, F., and Zucker, S. W. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the national academy of sciences*, 102(21):7426–7431, 2005.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, 2016.

Deshpande, Y., Sen, S., Montanari, A., and Mossel, E. Contextual stochastic block models. In *NeurIPS*, 2018.

Desquesnes, X., Elmoataz, A., and Lézoray, O. Eikonal equation adaptation on weighted graphs: fast geometric diffusion process for local and non-local image and data processing. *Journal of Mathematical Imaging and Vision*, 46(2):238–257, 2013.

Di Giovanni, F., Rowbottom, J., Chamberlain, B. P., Markovich, T., and Bronstein, M. M. Graph neural networks as gradient flows. *arXiv preprint arXiv:2206.10991*, 2022.

Dormand, J. and Prince, P. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

Elmoataz, A., Lezoray, O., and Bougleux, S. Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing. *IEEE transactions on Image Processing*, 17(7):1047–1060, 2008.

Fisher, R. A. The wave of advance of advantageous genes. *Annals of eugenics*, 7(4):355–369, 1937.

Freidlin, M. and Sheu, S.-J. Diffusion processes on graphs: stochastic differential equations, large deviation principle. *Probability theory and related fields*, 116(2):181–220, 2000.

Freidlin, M. I. and Wentzell, A. D. Diffusion processes on graphs and the averaging principle. *The Annals of probability*, pp. 2215–2245, 1993.

Gaudelet, T., Day, B., Jamasb, A. R., Soman, J., Regep, C., Liu, G., Hayter, J. B., Vickers, R., Roberts, C., Tang, J., et al. Utilizing graph machine learning within drug discovery and development. *Briefings in bioinformatics*, 22(6):bbab159, 2021.

Gilboa, G. and Osher, S. Nonlocal operators with applications to image processing. *Multiscale Modeling & Simulation*, 7(3):1005–1028, 2009.

Gilding, B. H. and Kersner, R. *Travelling waves in nonlinear diffusion-convection reaction*, volume 60. Springer Science & Business Media, 2004.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, 2017.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

He, M., Wei, Z., Huang, Z., and Xu, H. BernNet: Learning arbitrary graph spectral filters via bernstein approximation. In *NeurIPS*, 2021.

Huang, W., Zhang, T., Rong, Y., and Huang, J. Adaptive sampling towards fast graph representation learning. In *NeurIPS*, 2018.

Hwang, J., Choi, J., Choi, H., Lee, K., Lee, D., and Park, N. Climate modeling with neural diffusion equations. In *ICDM*, pp. 230–239, 2021.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Li, G., Muller, M., Thabet, A., and Ghanem, B. DeepGCNs: Can gcns go as deep as cnns? In *ICCV*, 2019.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018.

Li, X., Zhu, R., Cheng, Y., Shan, C., Luo, S., Li, D., and Qian, W. Finding global homophily in graph neural networks when meeting heterophily. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *ICML*, volume 162, pp. 13242–13256, 2022.

Li, Y., Jin, W., Xu, H., and Tang, J. Deeprobust: a platform for adversarial attacks and defenses. In *AAAI*, pp. 16078–16080, 2021.

Lim, D., Hohne, F., Li, X., Huang, S. L., Gupta, V., Bhalerao, O., and Lim, S. N. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *NeurIPS*, volume 34, pp. 20887–20902. Curran Associates, Inc., 2021.

Liu, M., Gao, H., and Ji, S. Towards deeper graph neural networks. In *KDD*, pp. 338–348, 2020.

Luan, S., Hua, C., Lu, Q., Zhu, J., Zhao, M., Zhang, S., Chang, X.-W., and Precup, D. Revisiting heterophily for graph neural networks. In *NeurIPS*, 2022.

Lyons, T., Caruana, M., and Lévy, T. *Differential Equations Driven by Rough Paths*. Springer, 2004. École D'Eté de Probabilités de Saint-Flour XXXIV - 2004.

McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, pp. 5425–5434, 2017.

Nt, H. and Maehara, T. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2020.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. In *ICLR*, 2020.

Poli, M., Massaroli, S., Park, J., Yamashita, A., Asama, H., and Park, J. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

Rozemberczki, B., Allen, C., and Sarkar, R. Multi-Scale Attributed Node Embedding. *Journal of Complex Networks*, 9(2), 2021.

Rusch, T. K., Chamberlain, B., Rowbottom, J., Mishra, S., and Bronstein, M. Graph-coupled oscillator networks. In *ICML*, volume 162, pp. 18888–18909, 2022.

Rusch, T. K., Bronstein, M. M., and Mishra, S. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv: Arxiv-2303.10993*, 2023.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective Classification in Network Data. *AI Magazine*, 29(3):93, September 2008.

Suresh, S., Budde, V., Neville, J., Li, P., and Ma, J. Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. In *KDD*, pp. 1541–1551, 2021.

Tang, J., Sun, J., Wang, C., and Yang, Z. Social influence analysis in large-scale networks. In *KDD*, pp. 807–816, 2009.

Thorpe, M., Nguyen, T. M., Xia, H., Strohmer, T., Bertozzi, A., Osher, S., and Wang, B. GRAND++: Graph neural diffusion with a source term. In *ICLR*, 2022.

Tiezzi, M., Marra, G., Melacci, S., and Maggini, M. Deep constraint-based propagation in graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(2):727–739, 2021.

Turing, A. The chemical basis of morphogenesis. *Phil. Trans. R. Soc. Lond. B*, 1952.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. In *ICLR*, 2018.

Wang, Y., Wang, Y., Yang, J., and Lin, Z. Dissecting the diffusion process in linear graph convolutional networks. In *NeurIPS*, 2021.

Wang, Y., Yi, K., Liu, X., Wang, Y. G., and Jin, S. ACMP: Allen-cahn message passing for graph neural networks with particle phase transition. In *ICLR*, 2023.

Wu, F., Zhang, T., de Souza, A. H., Fifty, C., Yu, T., and Weinberger, K. Q. Simplifying graph convolutional networks. In *ICML*, 2019.

Xhonneux, L.-P. A. C., Qu, M., and Tang, J. Continuous graph neural networks. In *ICML*, 2020.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *ICML*, pp. 5453–5462, 2018.

Yan, Y., Hashemi, M., Swersky, K., Yang, Y., and Koutra, D. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *ICDM*, 2022.

Yang, Z., Cohen, W. W., and Salakhutdinov, R. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, 2018.

Zhao, L. and Akoglu, L. PairNorm: Tackling oversmoothing in gnns. In *ICLR*, 2020.

Zhu, H. and Koniusz, P. Simple spectral graph convolution. In *ICLR*, 2020.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. In *NeurIPS*, 2020.

A. Full Ranking of Table 2

We show the average ranking/accuracy and the Olympic ranking of all methods in Table 7. Our methods occupy all the top-4 positions.

Table 7. The average ranking/accuracy and the Olympic ranking in the 9 real-world datasets. Methods are sorted by Avg. Ranking.

Method	Average		Olympic Ranking		
	Ranking	Accuracy	Gold	Silver	Bronze
GREAD-BS	1.56	76.64	5	4	0
GREAD-FB*	6.72	74.51	0	0	3
GREAD-F	7.50	74.13	1	1	1
GREAD-FB	7.89	74.48	0	0	1
GloGNN	8.17	74.99	0	0	1
GREAD-AC	8.50	73.71	1	0	0
ACM-GCN	8.67	74.92	0	1	0
GGCN	9.50	75.05	0	1	0
GREAD-Z	9.56	73.45	0	0	0
GREAD-ST	9.83	72.93	0	1	2
Sheaf	10.33	75.06	1	1	0
GRAFF	11.28	74.92	0	0	2
WRGAT	13.61	73.02	0	0	0
GCNII	14.56	70.16	0	0	0
H2GCN	16.00	71.33	0	0	0
BLEND	16.11	71.79	0	0	0
ACMP	19.44	71.98	0	0	0
GCON-GCN	20.17	69.46	0	0	1
LINKX	21.44	71.11	1	0	0
FA-GCN	21.89	69.86	0	0	0
GRAND	22.28	68.94	0	0	0
GCON-GAT	22.89	68.96	0	0	0
GPR-GNN	23.17	67.45	0	0	0
GraphSAGE	23.56	69.50	0	0	0
Geom-GCN	23.83	63.87	0	0	0
Mixhop	24.50	68.10	0	0	0
GCN	24.61	62.77	0	0	0
GDE	24.67	67.40	0	0	0
MLP	25.61	66.26	0	0	0
CGNN	26.11	63.96	0	0	0
ChebNet	26.56	67.98	0	0	0
GAT	28.22	60.23	0	0	0
PairNorm	28.56	61.68	0	0	0
JKNet	30.06	60.77	0	0	0
SGC	32.56	58.34	0	0	0

B. Full Result of Table 4

We reported only the selected highly-performing 12 baselines in Table 4. We now report all the detailed results of the tested 28 baselines in Table 8.

Table 8. Results on real-world datasets: mean ± std. dev. accuracy for 10 different data splits. We show the best three methods in red (first), blue (second), and purple (third).

Dataset	Texas	Wisconsin	Cornell	Film	Squirrel	Chameleon	Cora	Citeseer	PubMed
MLP	80.81±4.75	85.29±3.31	81.89±6.40	36.53±0.70	28.77±1.56	46.21±2.99	87.16±0.37	74.02±1.90	75.69±2.00
GCN	55.14±5.16	51.76±3.06	60.54±5.30	27.32±1.10	53.43±2.01	64.82±2.24	86.98±1.27	76.50±1.36	88.42±0.50
ChebNet	78.37±6.04	79.02±3.18	75.68±6.94	34.13±1.09	36.43±1.17	58.64±1.64	85.45±1.58	75.07±1.25	89.00±0.46
GAT	52.16±6.63	49.41±4.09	61.89±5.05	27.44±0.89	40.72±1.55	60.26±2.50	86.33±0.48	76.55±1.23	87.30±1.10
GraphSAGE	82.43±6.14	81.18±5.56	75.95±5.01	34.23±0.99	41.61±0.74	58.73±1.68	86.90±1.04	76.04±1.30	88.45±0.50
SGC	58.10±4.20	55.29±4.28	60.00±3.59	27.20±1.52	33.00±1.97	42.45±3.82	86.12±1.44	76.01±1.31	86.90±1.32
MixHop	77.84±7.73	75.88±4.90	73.51±6.34	32.22±2.34	43.80±1.48	60.50±2.53	87.61±0.85	76.26±1.33	85.31±0.61
Geom-GCN	66.76±2.72	64.51±3.66	60.54±3.67	31.59±1.15	38.15±0.92	60.00±2.81	85.35±1.57	78.02±1.15	89.95±0.47
FA-GCN	82.43±6.89	82.94±7.95	79.19±9.79	34.87±1.25	42.59±0.79	55.22±3.19	87.21±1.43	76.87±1.56	87.45±0.61
GPR-GNN	78.38±4.36	82.94±4.21	80.27±8.11	34.63±1.22	31.61±1.24	46.58±1.84	87.95±1.18	77.13±1.67	87.54±0.38
H2GCN	84.86±7.23	87.65±4.98	82.70±5.28	35.70±1.00	36.48±1.86	60.11±2.15	87.87±1.20	77.11±1.57	89.49±0.38
WRGAT	83.62±5.50	86.98±3.78	81.62±3.90	36.53±0.77	48.85±0.78	65.24±0.87	88.20±2.26	76.81±1.89	89.29±0.38
GGCN	84.86±4.55	86.86±3.29	85.68±6.63	37.54±1.56	55.17±1.58	71.14±1.84	87.95±1.05	77.14±1.45	89.15±0.37
LINKX	74.60±8.37	75.49±5.72	77.84±5.81	36.10±1.55	61.81±1.80	68.42±1.38	84.64±1.13	73.19±0.99	87.86±0.77
GloGNN	84.32±4.15	87.06±3.53	83.51±4.26	37.35±1.30	57.54±1.39	69.78±2.42	88.31±1.13	77.41±1.65	89.62±0.35
ACM-GCN	87.84±4.40	88.43±3.22	85.14±6.07	36.28±1.09	54.40±1.88	66.93±1.85	87.91±0.95	77.32±1.70	90.00±0.52
PairNorm	60.27±4.34	48.43±6.14	58.92±3.15	27.40±1.24	50.44±2.04	62.74±2.82	85.79±1.01	73.59±1.47	87.53±0.44
JKNet	62.70±8.34	53.14±5.22	59.72±4.60	29.25±1.37	39.78±1.72	52.63±3.90	86.48±1.04	75.99±1.28	87.23±0.55
GCNII	77.57±3.83	80.39±3.40	77.86±3.79	37.44±1.30	38.47±1.58	63.86±3.04	88.37±1.25	77.33±1.48	90.15±0.43
GCON-GCN	85.40±4.20	87.80±3.30	84.30±4.80	34.65±0.61	33.30±1.57	48.08±2.16	87.40±1.82	76.46±1.70	87.71±0.35
GCON-GAT	82.20±4.70	85.70±3.60	83.20±7.00	35.85±0.84	34.45±1.08	48.31±1.53	86.96±1.73	76.20±2.12	87.73±0.41
CGNN	71.35±4.05	74.31±7.26	66.22±7.69	35.95±0.86	29.24±1.09	46.89±1.66	87.10±1.35	76.91±1.81	87.70±0.49
GDE	74.05±6.96	79.80±5.62	82.43±7.07	35.36±1.31	35.94±1.91	47.76±2.08	87.22±1.41	76.21±2.11	87.80±0.38
GRAND	75.68±7.25	79.41±3.64	82.16±7.09	35.62±1.01	40.05±1.50	54.67±2.54	87.36±0.96	76.46±1.77	89.02±0.51
BLEND	83.24±4.65	84.12±3.56	85.95±6.82	35.63±1.01	43.06±1.39	60.11±2.09	88.09±1.22	76.63±1.60	89.24±0.42
ACMP	86.20±0.30	86.10±0.40	85.40±0.70	34.44±4.44	52.65±2.23	52.63±2.28	86.38±3.79	76.52±1.84	87.54±0.57
Sheaf	85.05±5.51	89.41±4.74	84.86±4.71	37.81±1.15	56.34±1.32	68.04±1.58	86.90±1.13	76.70±1.57	89.49±0.40
GRAFF	88.38±4.53	87.45±2.94	83.24±6.49	36.09±0.81	54.52±1.37	71.08±1.75	87.61±0.97	76.92±1.70	88.95±0.52
GREAD-BS	88.92±3.72	89.41±3.30	86.49±7.15	37.90±1.17	59.22±1.44	71.38±1.53	88.57±0.66	77.60±1.81	90.23±0.55
GREAD-F	89.73±4.49	86.47±4.84	86.49±5.13	36.72±0.66	46.16±1.44	65.20±1.40	88.39±0.91	77.40±1.54	90.09±0.31
GREAD-AC	85.95±2.65	86.08±3.56	87.03±4.95	37.21±1.10	45.10±2.11	65.09±1.08	88.29±0.67	77.38±1.53	90.10±0.36
GREAD-Z	87.30±5.68	86.29±4.32	85.68±5.41	37.01±1.11	46.25±1.72	62.70±2.30	88.31±1.10	77.39±1.90	90.11±0.27
GREAD-ST	81.08±5.67	86.67±3.01	86.22±5.98	37.66±0.90	45.83±1.40	63.03±1.32	88.47±1.19	77.25±1.47	90.13±0.36
GREAD-FB	86.76±5.05	87.65±3.17	86.22±5.85	37.40±0.55	50.83±2.27	66.05±1.21	88.01±1.34	77.28±1.73	90.07±0.45
GREAD-FB*	87.03±3.97	88.04±1.63	85.95±5.64	37.70±0.51	50.57±1.52	65.83±1.10	88.01±0.80	77.42±1.93	90.08±0.46

C. Full Derivation of Eq. (12)

We show the omitted intermediate derivation steps of Eq. (12).

$$
\begin{aligned}
\mathbf{H}(t+h) &= \mathbf{B}(t+h) + \tilde{\mathbf{L}}(\mathbf{B}(t+h)), \\
&\Rightarrow \tilde{\mathbf{A}}\mathbf{H}(t) + \tilde{\mathbf{L}}(\tilde{\mathbf{A}}\mathbf{H}(t)), \\
&\Rightarrow \tilde{\mathbf{A}}\mathbf{H}(t) + (\mathbf{I} - \tilde{\mathbf{A}})\tilde{\mathbf{A}}\mathbf{H}(t), \\
&\Rightarrow 2\tilde{\mathbf{A}}\mathbf{H}(t) - \tilde{\mathbf{A}}^2\mathbf{H}(t), \\
&\Rightarrow (2\mathbf{I} - \tilde{\mathbf{A}})\tilde{\mathbf{A}}\mathbf{H}(t), \\
&\Rightarrow (\mathbf{I} + \tilde{\mathbf{L}})(\mathbf{I} - \tilde{\mathbf{L}})\mathbf{H}(t), \\
&\Rightarrow \mathbf{H}(t) - \tilde{\mathbf{L}}^2\mathbf{H}(t), \\
&\Rightarrow \mathbf{H}(t) - (\mathbf{I} - \tilde{\mathbf{A}})^2\mathbf{H}(t), \\
&\Rightarrow \mathbf{H}(t) - (\mathbf{I} - \tilde{\mathbf{A}})\mathbf{H}(t) + (\tilde{\mathbf{A}} - \tilde{\mathbf{A}}^2)\mathbf{H}(t), \\
&\Rightarrow \mathbf{H}(t) - \tilde{\mathbf{L}}\mathbf{H}(t) + (\tilde{\mathbf{A}} - \tilde{\mathbf{A}}^2)\mathbf{H}(t).
\end{aligned}
$$

D. Computational Complexity

The space complexity of GREAD is dominated by evaluating the soft adjacency matrix in Eq. (9), which is $\mathcal{O}(|\mathcal{E}|\dim(\mathbf{H}))$, where $|\mathcal{E}|$ is the number of edges and $\dim(\mathbf{H})$ is the size of hidden dimension.

We also analyze the time complexity of the reaction-diffusion layer in Eq. (7). Our proposed model has different complexity depending on the reaction term r in Eq. (10).

If we set the adjacency matrix and β to OA and SC respectively, the time complexity of the one-step GREAD-BS computation becomes $\mathcal{O}(n_\tau(|\mathcal{E}| + |\mathcal{E}_2|)\dim(\mathbf{H}) + |\mathcal{E}|d_{\max})$, where n_τ, $|\mathcal{V}|$, and $d_{\max}$ are the number of steps in $[0, T]$, the number of nodes, and the maximum degree of all nodes, respectively. Given that $\mathbf{A}$ is sparse, we can calculate $\mathbf{A}^2$ in $\mathcal{O}(|\mathcal{E}|d_{\max})$ because $d_{\max}$ is equal to the maximum number of non-zeroes in any row of $\mathbf{A}$. The sparse matrix multiplication of $\mathbf{A}^2\mathbf{H}(t)$ takes $\mathcal{O}(|\mathcal{E}_2|d_{\max})$, where $|\mathcal{E}_2| = \frac{1}{2}\sum_{v\in\mathcal{V}}|\mathcal{N}_2|(v)$. The computational complexity of the one-step GREAD-F computation is $\mathcal{O}(n_\tau(|\mathcal{E}| + \dim(\mathbf{H})^k))\dim(\mathbf{H}))$, where $k = 1$. In the case of GREAD-AC and GREAD-Z, their k values are 2 and 3, respectively. The computational complexities of the one-step GREAD-ST, GREAD-FB, and GREAD-FB* are $\mathcal{O}(n_\tau|\mathcal{E}|\dim(\mathbf{H}) + |\mathcal{E}|d_{\max})$.

E. Additional Details for Experiments

E.1. Details of Datasets

Real-world Datasets For the experiment with real-world datasets in Table 4, we consider both the heterophilic and homophilic datasets. They can be distinguished based on the homophily level. We employ the homophily ratio, defined by (Pei et al., 2020), to distinguish high or low homophily/heterophily graphs:

$$
\text{Homophily ratio} = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}}\frac{\sum_{u\in\mathcal{N}_v}(y_u = y_v)}{|\mathcal{N}_v|}. \tag{19}
$$

A high homophily ratio means that neighbors tend to be in an identical class. Some dataset statistics are given in Table 3. The 9 real-world datasets we consider are as follows:

- Chameleon and Squirrel are subgraphs of web pages in Wikipedia (Rozemberczki et al., 2021). The node in Wikipedia graphs represent web pages, the edge mean mutual links between pages, and the node feature corresponds to several informative nouns in the Wikipedia page. All nodes are classified into 5 categories based on the average monthly traffic.

- Film is a subgraph of the film-director-actor-writer network (Tang et al., 2009). Each node corresponds to an actor, an edge between two nodes denotes the co-occurrence relationship in a Wikipedia page, and the node feature corresponds to some keywords in the Wikipedia page. All nodes are classified into 5 categories according to the type of actors.

- Cornell, Texas, and Wisconsin are three subsets of the WebKB dataset collected by CMU, having many links between web pages of the universities. In these networks, nodes represent web pages, edges are hyperlinks between them, and node features are the bag-of-words representation of web pages. All nodes are classified into 5 categories: student, project, course, staff, and faculty.

- Cora (McCallum et al., 2000), Citeseer (Sen et al., 2008), and Pubmed (Yang et al., 2016) are among the most widely used benchmark datasets for the semi-supervised node classification. These are citation networks, where nodes, edges, features, and labels respectively correspond to papers, undirected paper citations, the bag-of-words representations of papers, and the academic topics of papers.

The Synthetic Cora Network   The synthetic Cora dataset is provided by (Zhu et al., 2020). They generate graphs for a target homophily level using a modified preferential attachment process. Nodes, edges, and features are sampled from Cora to create a synthetic graph with a desired homogeneity and feature/label distribution. In Table 9, we summarize the properties of the synthetic Cora networks we used.

Table 9. The detailed information of the synthetic Cora. All levels of homophily have the same number of features (1,433), nodes (1,480), edges (5,936), and classes (5).

Homophily	Avg. Degree	Max. Degree	Min. Degree
0.0	3.98	84.33	1.67
0.1	3.98	71.33	2.00
0.2	3.98	73.33	1.67
0.3	3.98	70.00	2.00
0.4	3.98	77.67	2.00
0.5	3.98	76.33	2.00
0.6	3.98	76.00	1.67
0.7	3.98	67.67	2.00
0.8	3.98	58.00	1.67
0.9	3.98	58.00	1.67
1.0	3.98	51.00	2.00

The cSBM Synthetic Network   For Fig. 5, we use cSBM (Deshpande et al., 2018) to generate synthetic networks. cSBM generates Gaussian random vectors as node features on top of the classical SBM. The synthetic graph has 100 nodes with 2 classes and two-dimensional features sampled from a normal distribution with $\sigma = 2$, $\mu_1 = -0.5$, and $\mu_2 = 0.5$. The nodes are randomly connected with a probability of $p = 0.9$ if they are in the same class and $p = 0.1$ otherwise.

E.2. Details of Experimental Settings

Evolution of the Dirichlet Energy   We use the random graphs generated by cSBM to show the capability of GREAD to alleviate oversmoothing. In the case of GREAD, we run without any hyperparamerter search but list the full hyperparameter list we used in Table 10.

Comparison with Various Homophily Rate   To compare the performance in various homophily rates, we use the synthetic Cora network. We run the experiment with 3 fixed train/valid/test splits and report the mean and the standard deviation of accuracy accordingly. In Table 11, we list the hyperparameter range we consider.

Node Classification on Real-world Datasets   The following software and hardware environments were used for all experiments: UBUNTU 18.04 LTS, PYTHON 3.9.12, PYTORCH 1.11.0, PYTORCH GEOMETRIC 2.0.4, TORCHDIFFEQ 0.2.3, NUMPY 1.22.4, SCIPY 1.8.1, MATPLOTLIB 2.2.3, CUDA 11.3, and NVIDIA Driver 465.19, and i9 CPU, and NVIDIA RTX 3090. We performed 10 repetitions on the train/valid/test splits taken from (Pei et al., 2020) and strictly followed their evaluation protocol. For all data sets, we used the largest connectivity component (LCC) except for Citeseer. We use the dropout only in the encoder network and the output layer. We refer to the dropout in the encoder as 'input dropout' and the dropout in the output layer as 'dropout'.

We fine-tune our model within the hyperparameter search space in Table 12. Our hyperparameter search used the method of W&B Sweeps (Biewald, 2020) with a standard random search with 500 counts. We introduce the best hyperparameter configuration in Tables 13 to 16.

Table 10. Hyperparameter for the cSBM synthetic network

Hyperparameters	Value
epochs	100
adjacency matrix	OA
α	SC
β	VC
learning rate	0.001
weight decay	5×10^{-4}
dropout	0.0
input dropout	0.5
dim($\mathbf{H}$)	2
step size τ	1.0
time T	40
ODE solver	Euler

Table 11. Hyperparameter search space for the synthetic Cora network

Hyperparameters	Search Space
epochs	100
adjacency matrix	{OA, SA}
α	{SC, VC}
β	{SC, VC}
learning rate	{0.001, 0.002, 0.0025, 0.005, 0.01}
weight decay	{0.01, 0.001, 0.0005, 0.0001}
dropout	0.35
input dropout	0.5
dim($\mathbf{H}$)	64
step size τ	{0.1, 0.5, 1.0}
time T	{1, 2, 3, 4}
ODE solver	Euler

Table 12. Hyperparameter search space for real-world datasets

Hyperparameters	Search Space
epochs	200
adjacency matrix	{OA, SA}
α	{SC, VC}
β	{SC, VC}
learning rate	$[1 \times 10^{-3}, 6 \times 10^{-2}]$
weight decay	$[0, 3 \times 10^{-2}]$
dropout	[0, 0.6]
input dropout	[0, 0.6]
dim($\mathbf{H}$)	{32, 64, 128, 256}
step size τ	[0.1, 1.5]
time T	[0.1, 6.0]
ODE solver	{Euler, RK4, DOPRI5}

Table 13. Best hyperparameters of GREAD-BS

Hyperparameters	Texas	Wisconsin	Cornell	Film	Squirrel	Chameleon	Cora	Citeseer	PubMed
adjacency matrix	OA	SA	SA	SA	OA	SA	SA	SA	SA
α	SC	SC	VC	SC	VC	VC	VC	SC	VC
β	VC	VC	VC	VC	VC	VC	SC	SC	SC
learning rate	0.0100	0.0154	0.0082	0.0079	0.0171	0.0068	0.0105	0.0024	0.0108
weight decay	0.0247	0.0090	0.0280	0.0014	0.0000	0.0000	0.0060	0.0146	0.0005
input dropout	0.47	0.54	0.49	0.42	0.52	0.68	0.53	0.50	0.36
dropout	0.48	0.48	0.32	0.65	0.09	0.05	0.45	0.47	0.26
dim($\mathbf{H}$)	128	256	128	64	256	256	64	128	64
step size τ	1.0	0.25	0.2	0.1	0.75	1.5	0.25	0.5	0.8
time T	1.46	0.75	0.12	0.31	5.70	1.71	3.49	2.35	1.74
ODE solver	Euler	RK4	RK4	RK4	Euler	Euler	RK4	RK4	RK4

Table 14. Best hyperparameters of GREAD-F

Hyperparameters	Texas	Wisconsin	Cornell	Film	Squirrel	Chameleon	Cora	Citeseer	PubMed
adjacency matrix	SA	SA	SA	SA	SA	SA	SA	SA	SA
α	VC	SC	VC	SC	VC	SC	SC	SC	VC
β	VC	VC	VC	VC	VC	VC	SC	VC	VC
learning rate	0.0113	0.0094	0.0092	0.0068	0.0054	0.0101	0.0048	0.0013	0.0120
weight decay	0.0079	0.0057	0.0263	0.0006	0.0011	0.0015	0.0370	0.0041	0.0003
input dropout	0.46	0.41	0.46	0.48	0.48	0.50	0.50	0.50	0.36
dropout	0.38	0.05	0.31	0.48	0.36	0.24	0.35	0.51	0.25
dim($\mathbf{H}$)	256	64	256	128	128	256	32	256	128
step size τ	1.0	0.1	1.0	0.75	1.0	1.0	0.2	0.9	1
time T	1.26	0.12	1.0	1.14	2.23	1.0	2.27	1.86	1.44
ODE solver	Euler	RK4	Euler	RK4	Euler	RK4	Euler	RK4	RK4

Table 15. Best hyperparameters of GREAD-AC

Hyperparameters	Texas	Wisconsin	Cornell	Film	Squirrel	Chameleon	Cora	Citeseer	PubMed
adjacency matrix	SA	SA	SA	SA	SA	SA	SA	SA	SA
α	VC	VC	SC	SC	SC	SC	SC	SC	VC
β	VC	VC	VC	SC	VC	VC	VC	VC	VC
learning rate	0.0070	0.0083	0.0084	0.0027	0.0025	0.0038	0.0039	0.0029	0.0124
weight decay	0.0136	0.0081	0.0311	0.0001	0.0020	0.0007	0.0469	0.0140	0.0006
input dropout	0.40	0.45	0.49	0.46	0.52	0.52	0.40	0.47	0.30
dropout	0.30	0.20	0.29	0.48	0.28	0.35	0.40	0.49	0.26
dim($\mathbf{H}$)	256	128	128	128	128	256	128	64	128
step size τ	1.0	0.5	0.75	1.0	1.0	1.0	0.1	0.9	1.0
time T	1.36	0.20	0.18	1.06	1.98	2.0	3.52	2.78	1.65
ODE solver	Euler	RK4	RK4	Euler	Euler	RK4	Euler	RK4	RK4

Table 16. Best hyperparameters of GREAD-Z

Hyperparameters	Texas	Wisconsin	Cornell	Film	Squirrel	Chameleon	Cora	Citeseer	PubMed
adjacency matrix	OA	SA	SA	SA	SA	SA	SA	SA	SA
α	VC	SC	SC	SC	VC	VC	VC	VC	VC
β	SC	VC	SC	VC	VC	VC	SC	VC	VC
learning rate	0.0088	0.0046	0.0048	0.0023	0.0099	0.0111	0.0045	0.0027	0.0091
weight decay	0.0462	0.0086	0.0435	0.0011	0.0007	0.0012	0.0050	0.0145	0.0004
input dropout	0.48	0.45	0.4272	0.48	0.53	0.45	0.4	0.50	0.37
dropout	0.46	0.18	0.29	0.48	0.44	0.31	0.2	0.49	0.22
dim($\mathbf{H}$)	256	128	256	64	128	256	64	64	64
step size τ	1.2	0.4	0.2	0.2	1.0	1.0	0.1	0.8	0.8
time T	1.2	0.11	0.13	0.75	2.71	1.0	3.55	2.01	1.12
ODE solver	RK4	RK4	RK4	RK4	RK4	RK4	RK4	RK4	RK4

Table 17. Best hyperparameters of GREAD-ST

Hyperparameters	Texas	Wisconsin	Cornell	Film	Squirrel	Chameleon	Cora	Citeseer	PubMed
adjacency matrix	OA	SA	SA	SA	SA	SA	SA	SA	SA
α	SC	SC	SC	SC	VC	VC	SC	SC	SC
β	SC	VC	VC	SC	VC	VC	VC	SC	SC
learning rate	0.0200	0.0180	0.0050	0.0081	0.0538	0.0077	0.0074	0.0038	0.0108
weight decay	0.0295	0.0082	0.0275	0.0013	0.0000	0.0000	0.0086	0.0042	0.0004
input dropout	0.46	0.54	0.47	0.42	0.61	0.65	0.37	0.49	0.36
dropout	0.50	0.50	0.25	0.56	0.95	0.09	0.41	0.54	0.22
dim($\mathbf{H}$)	126	256	256	64	256	256	128	64	64
step size τ	0.5	0.5	0.25	0.7	1.0	1.0	0.1	0.6	0.9
time T	1.02	0.1	0.20	0.15	3.54	1.0	3.04	2.37	1.28
ODE solver	Euler	RK4	RK4	RK4	Euler	Euler	RK4	RK4	RK4

Table 18. Best hyperparameters of GREAD-FB

Hyperparameters	Texas	Wisconsin	Cornell	Film	Squirrel	Chameleon	Cora	Citeseer	PubMed
adjacency matrix	OA	SA	SA	SA	SA	SA	SA	OA	SA
α	VC	SC	SC	SC	VC	VC	SC	VC	VC
β	SC	VC	VC	VC	VC	VC	VC	SC	VC
learning rate	0.0016	0.0185	0.0050	0.0133	0.0090	0.0010	0.0064	0.0012	0.0102
weight decay	0.0055	0.0113	0.0283	0.0014	0.0000	0.0000	0.0091	0.0042	0.0004
input dropout	0.52	0.50	0.36	0.51	0.62	0.64	0.47	0.45	0.35
dropout	0.48	0.53	0.23	0.60	0.06	0.05	0.50	0.54	0.21
dim($\mathbf{H}$)	64	64	256	64	128	128	256	128	64
step size τ	1.5	0.7	1.0	0.6	0.25	0.25	0.5	0.6	0.2
time T	1.4	1.0	0.1	1.3	2.4	1.8	3.1	1.5	1.0
ODE solver	Euler	RK4	RK4	RK4	Euler	Euler	RK4	RK4	Euler

Table 19. Best hyperparameters of GREAD-FB*

Hyperparameters	Texas	Wisconsin	Cornell	Film	Squirrel	Chameleon	Cora	Citeseer	PubMed
adjacency matrix	OA	SA	SA	SA	SA	SA	SA	SA	SA
α	SC	SC	SC	SC	VC	VC	VC	VC	SC
β	VC	VC	SC	VC	VC	VC	VC	SC	VC
learning rate	0.0194	0.0195	0.0072	0.0144	0.0055	0.0095	0.0097	0.0020	0.0166
weight decay	0.0113	0.0142	0.0169	0.0010	0.0000	0.0000	0.0090	0.0048	0.0005
input dropout	0.45	0.51	0.36	0.52	0.63	0.65	0.50	0.57	0.35
dropout	0.52	0.48	0.19	0.59	0.05	0.14	0.39	0.39	0.22
dim($\mathbf{H}$)	64	64	128	64	128	128	64	64	128
step size τ	1.5	1.0	0.1	0.8	0.1	0.2	0.1	0.9	1.0
time T	1.4	1.0	0.2	1.9	2.0	1.5	3.3	1.7	1.4
ODE solver	Euler	Euler	Euler	RK4	Euler	Euler	Euler	RK4	RK4

F. Additional Experimental Results on Real-world Datasets

F.1. Ablation Studies

Tables 20 to 21 show the results of our additional ablation studies in the remaining datasets that are not reported in our main paper. In Table 20, SA outperforms OA in all the datasets except for Texas and Wisconsin. In the case of Texas and Wisconsin, SA performs worse than OA from time to time. In Table 21, we compare two types of β. β can be either a scalar parameter (SC) or a learnable vector parameter (VC). In almost cases, it shows better performance when the type of β is VC.

Table 20. Ablation study on soft adjacency matrix

Dataset	A	GREAD-BS	GREAD-F	GREAD-AC	GREAD-Z	GREAD-ST	GREAD-FB	GREAD-FB*
Texas	OA	**88.92**±**3.72**	86.49±4.69	85.41±5.16	**87.30**±**5.68**	**81.08**±**5.67**	**86.76**±**5.05**	**87.03**±**3.97**
	SA	85.41±2.76	**89.73**±**4.49**	**85.95**±**2.65**	86.49±3.20	80.00±6.23	84.41±4.22	85.14±5.57
Wisconsin	OA	87.45±3.53	**86.47**±**4.16**	**87.26**±**3.87**	86.28±3.62	85.88±3.26	85.13±4.13	85.42±4.51
	SA	**89.41**±**3.30**	**86.47**±**4.84**	85.69±5.04	**86.29**±**4.32**	**86.67**±**3.01**	**87.65**±**3.17**	**88.04**±**1.63**
Squirrel	OA	47.03±1.31	37.85±1.11	38.07±1.71	38.43±1.37	41.56±1.74	49.88±1.44	49.21±1.95
	SA	**59.22**±**1.44**	**46.16**±**1.44**	**45.10**±**2.11**	**46.25**±**1.72**	**45.83**±**1.40**	**50.83**±**2.27**	**50.57**±**1.52**
Chameleon	OA	67.79±1.91	59.80±1.54	58.93±1.92	54.45±2.29	56.05±1.28	62.41±1.99	62.63±1.64
	SA	**71.38**±**1.31**	**65.20**±**1.65**	**65.09**±**1.08**	**62.70**±**2.30**	**62.30**±**1.99**	**66.05**±**1.21**	**65.83**±**1.10**
Cora	OA	87.34±1.34	86.72±1.17	86.88±1.09	86.90±1.02	87.77±1.35	88.01±1.34	87.67±1.14
	SA	**88.57**±**0.66**	**88.39**±**0.91**	**88.29**±**0.67**	**88.31**±**1.10**	**88.47**±**1.19**	88.03±0.78	**88.01**±**0.80**
Citeseer	OA	77.33±1.74	76.29±1.74	76.73±1.52	76.69±1.97	76.57±1.29	**77.28**±**1.73**	77.38±1.79
	SA	**77.60**±**1.81**	**77.40**±**1.54**	**77.38**±**1.53**	**77.39**±**1.73**	**77.25**±**1.47**	77.09±1.73	**77.42**±**1.93**
Pubmed	OA	89.98±0.38	87.99±0.41	88.31±0.44	88.83±0.37	87.87±0.33	89.96±0.33	89.58±0.39
	SA	**90.23**±**0.55**	**90.09**±**0.31**	**90.10**±**0.36**	**90.11**±**0.27**	**90.13**±**0.36**	**90.07**±**0.45**	**90.08**±**0.46**

Table 21. Ablation study on β

Dataset	β	GREAD-BS	GREAD-F	GREAD-AC	GREAD-Z	GREAD-ST	GREAD-FB	GREAD-FB*
Cornell	SC	85.14±5.57	85.41±6.75	85.41±6.96	84.60±6.17	85.95±6.60	85.65±6.21	84.16±6.02
	VC	**86.49**±**7.15**	**86.49**±**5.13**	**87.03**±**4.95**	**85.68**±**5.41**	**86.22**±**5.98**	**86.22**±**5.85**	**85.95**±**5.64**
Film	SC	37.09±1.15	36.53±1.04	**37.21**±**1.10**	**37.01**±**1.11**	**37.66**±**0.90**	35.07±0.92	34.24±1.21
	VC	**37.90**±**1.17**	**37.20**±**1.26**	36.76±0.99	36.70±0.69	37.33±1.35	**37.40**±**0.55**	**37.70**±**0.51**
Squirrel	SC	42.74±1.34	44.88±1.62	39.61±1.69	40.33±2.06	43.41±1.61	40.59±1.14	40.15±1.66
	VC	**59.22**±**1.44**	**46.16**±**1.44**	**45.10**±**2.11**	**46.25**±**1.72**	**45.83**±**1.40**	**50.83**±**2.27**	**50.57**±**1.52**
Chameleon	SC	62.02±1.86	61.80±1.80	56.56±2.28	59.17±1.26	60.70±1.40	57.57±1.83	57.70±2.11
	VC	**71.38**±**1.31**	**65.20**±**1.65**	**65.09**±**1.08**	**62.70**±**2.30**	**62.30**±**1.99**	**66.05**±**1.21**	**65.83**±**1.10**
Cora	SC	87.45±1.08	88.07±0.96	88.01±0.85	88.13±0.40	88.35±1.32	87.75±1.24	86.68±0.88
	VC	**88.57**±**0.66**	**88.39**±**0.91**	**88.29**±**0.67**	**88.31**±**1.10**	**88.47**±**1.19**	**88.03**±**0.78**	**88.01**±**0.80**
Citeseer	SC	76.73±1.73	76.70±1.75	75.83±1.36	76.83±1.16	**77.25**±**1.47**	**77.28**±**1.73**	**77.42**±**1.93**
	VC	**77.60**±**1.81**	**77.40**±**1.54**	**77.38**±**1.53**	**77.39**±**1.73**	77.13±2.20	77.22±2.13	77.23±1.89
Pubmed	SC	89.96±0.42	87.51±0.44	88.76±0.45	90.04±0.26	**90.13**±**0.36**	89.90±0.47	89.99±0.24
	VC	**90.23**±**0.55**	**90.09**±**0.31**	**90.10**±**0.36**	**90.11**±**0.27**	90.10±0.41	**90.07**±**0.45**	**90.08**±**0.46**

F.2. Sensitivity Analyses

In Figs. 7 and 8, we show the findings of our sensitivity studies in the remaining datasets that are not disclosed in our main manuscript. GREAD-BS maintains performance even when T is increased, but GREAD-Z tends to show low performance in Texas, Cornell, and Film.
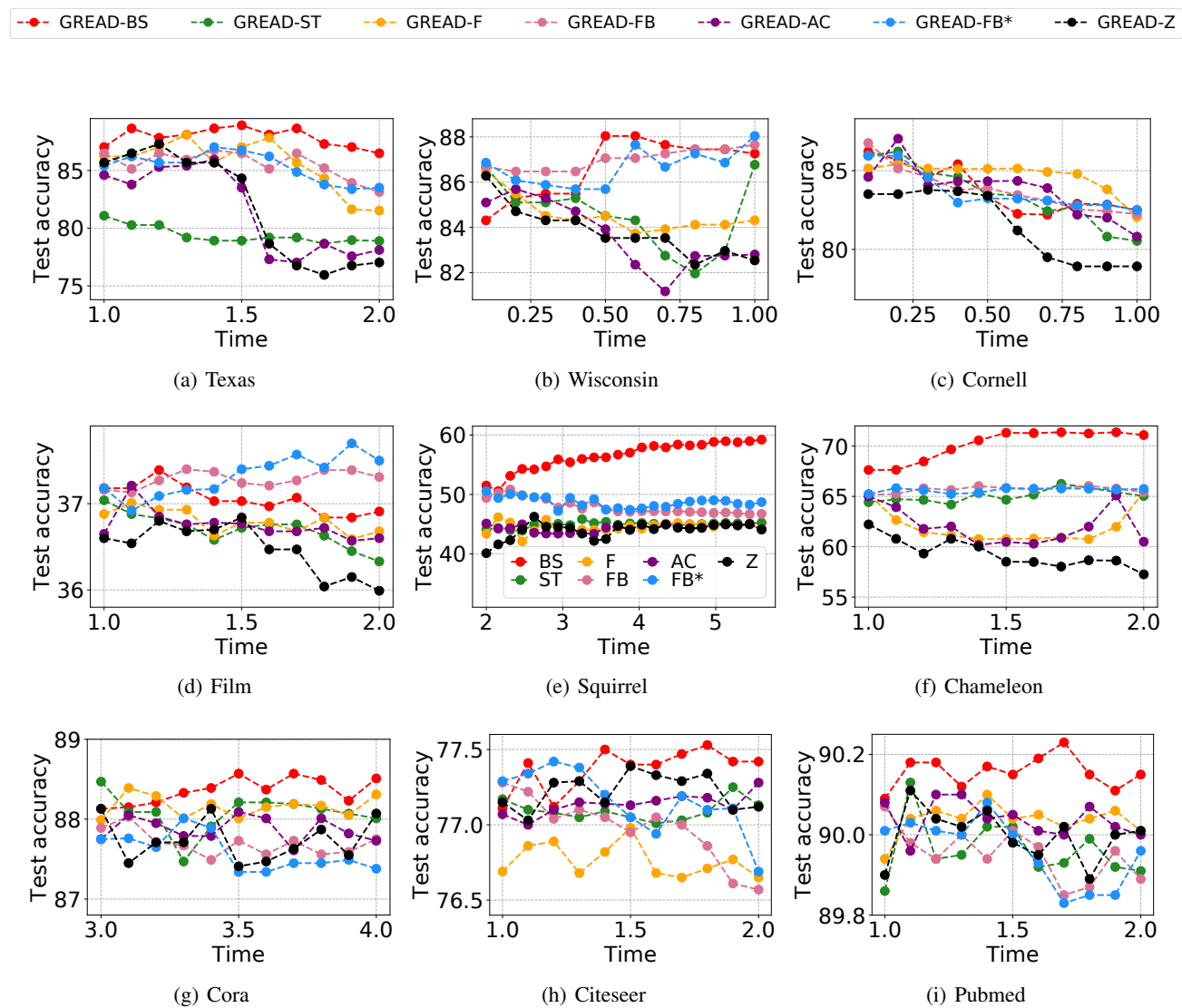

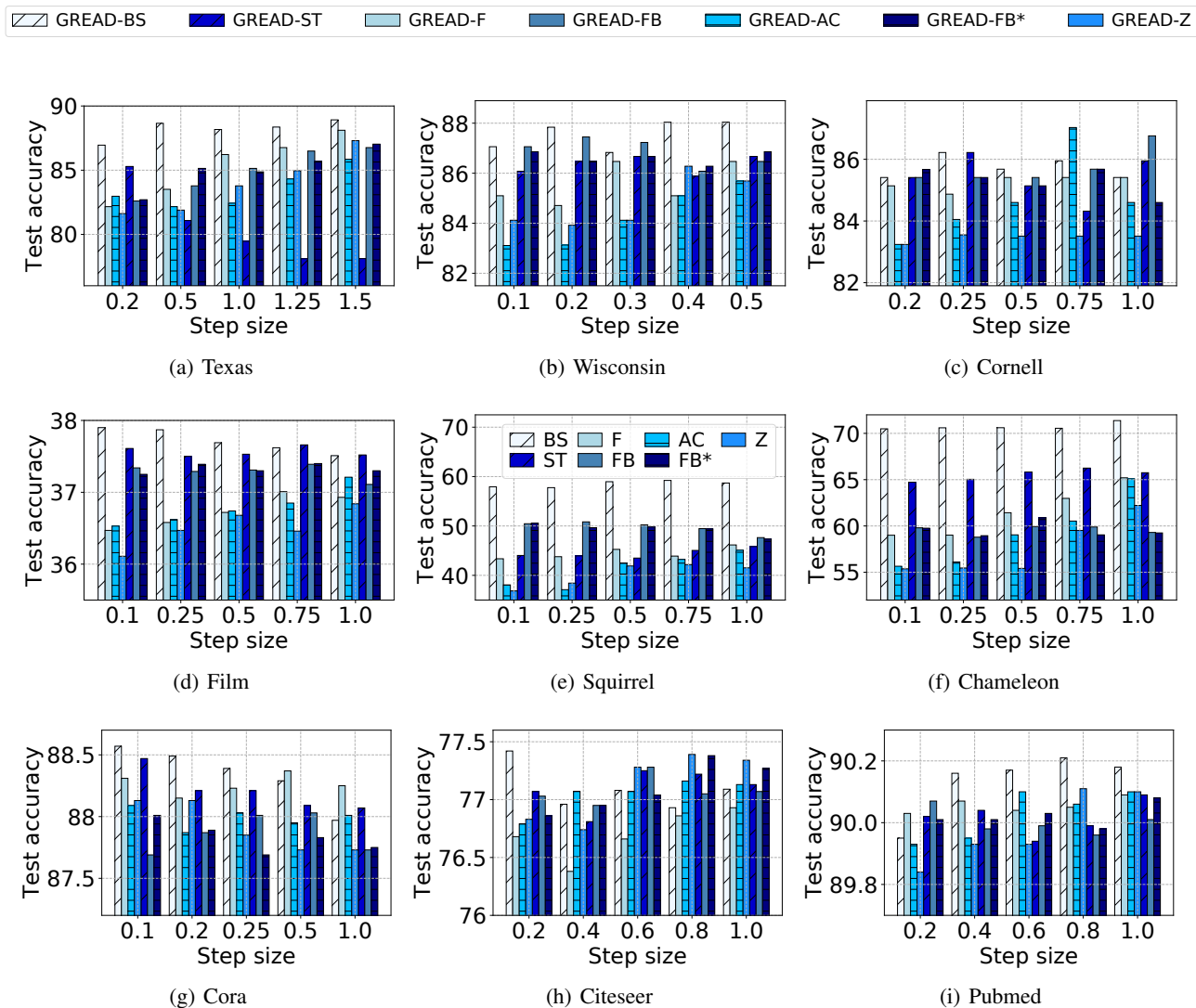*Figure 7.* Sensitivity to T

Figure 8. Sensitivity to step size

F.3. Training Time

We present the training time of GREAD and some selected baselines in Fig. 9. In general, our method's training time is little larger than those of the existing baselines because GREAD has an additional operation in its reaction term.

(a) Euler (b) RK4
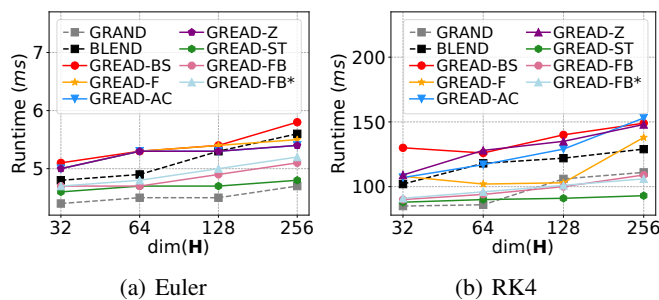
Figure 9. Average running time per epoch (ms) on Cora dataset when $T = 3$, $\tau = 1.0$, SC, and OA.

F.4. Visualizations

In order to show the effectiveness of our proposed model more intuitively, we further conduct visualization tasks for all datasets. We extract the output vector in the final layer of GREAD and visualize those vectors using t-SNE. Fig. 10 shows the visualization results on each dataset. Different colors mean different ground-truth classes.

(a) Texas

(b) Wisconsin

(c) Cornell

(d) Film

(e) Squirrel

(f) Chameleon

(g) Cora

(h) Citeseer

(i) Pubmed

Figure 10. Visualization of networks
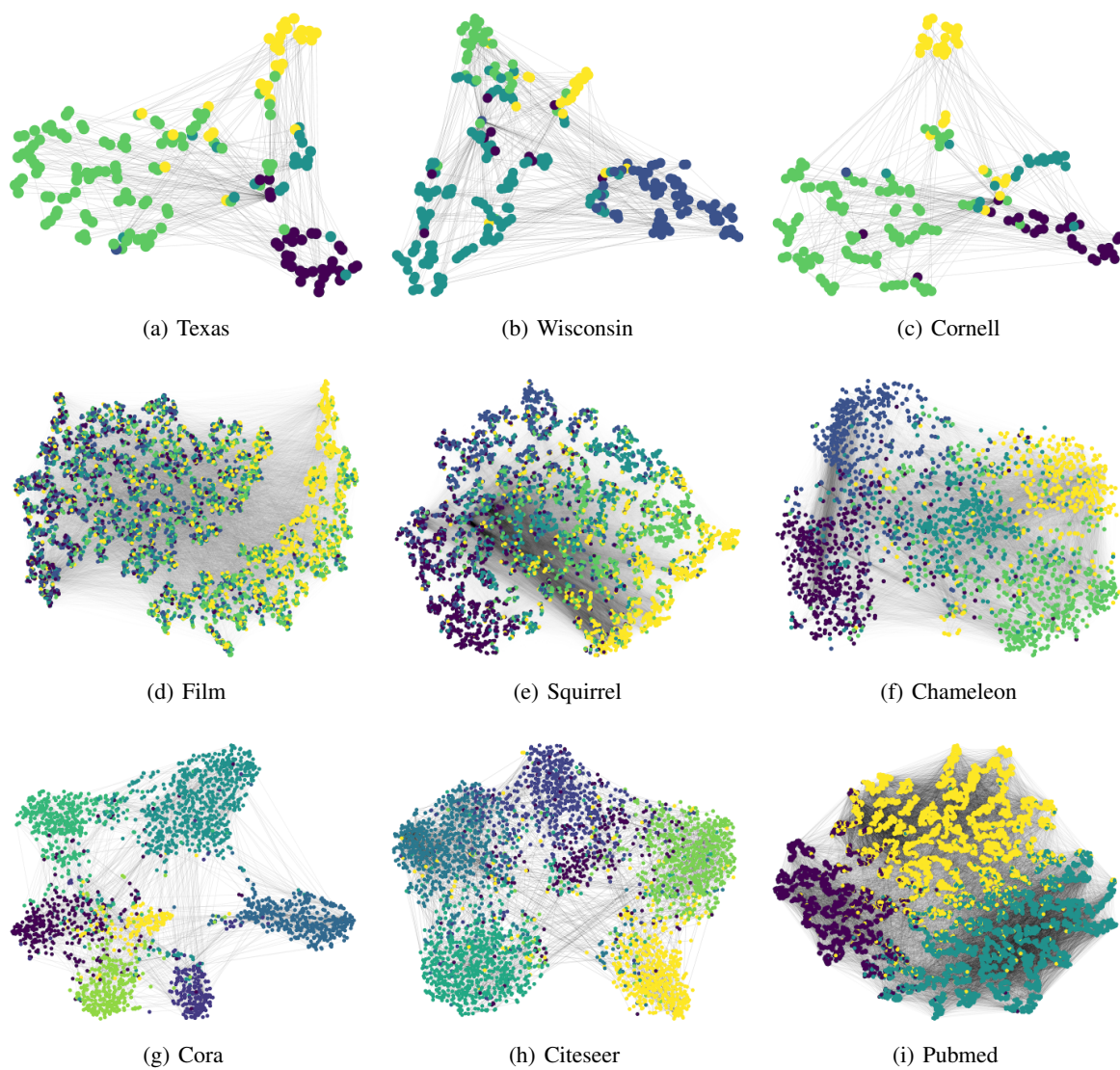
G. Additional Experimental Results on Synthetic Datasets

G.1. Ablation Studies on β

We perform the ablation study on β from the perspective of the Dirichlet energy. β can be either a scalar parameter (SC) or a learnable vector parameter (VC). In Fig. 11, we show the evolution of the Dirichlet energy on the synthetic random graph created from cSBM (Deshpande et al., 2018), and compare SC and VC for our proposed method. In the case of GREAD-F, GREAD-AC, GREAD-ST, and GREAD-FB, VC conserves more energy than SC, so the reaction term multiplied with β successfully mitigates the oversmoothing problem.

(a) GREAD-BS	(b) GREAD-F	(c) GREAD-AC	(d) GREAD-Z

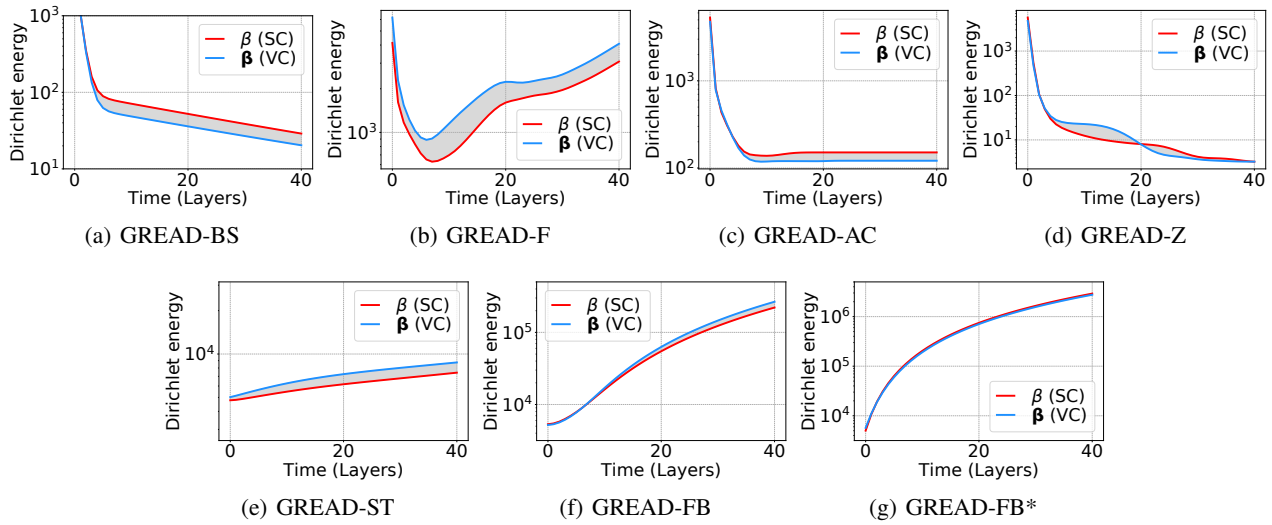| (e) GREAD-ST	(f) GREAD-FB	(g) GREAD-FB*

Figure 11. Evolution of the Dirichlet energy on the synthetic random graph. The Y-axis is the logarithmic Dirichlet energy in each layer's output given a GNN of 40 layers. The gray area is the Dirichlet energy difference between SC and VC.

H. Comparison with GRAND++ and GREAD-ST

The structures of GREAD-ST and GRAND++ are similar to each other since they commonly use a source term to alleviate the oversmoothing problem. However, there exists a key difference. First, GRAND++ is written as follows:

$$f(\mathbf{H}(t))_{\text{GRAND++}} := -\mathbf{L}\mathbf{H}(t) + \mathbf{C}(0), \tag{20}$$

where $\mathbf{C}(0)$ is a subset of $\mathbf{H}(0)$, which consists of only "trustworthy" nodes.

On the other hand, the formula of GREAD-ST is written as follows:

$$f(\mathbf{H}(t))_{\text{GREAD-ST}} := \alpha(-\mathbf{L}\mathbf{H}(t)) + \beta(\mathbf{C}(0)). \tag{21}$$

In our GREAD-ST, the full source term $\mathbf{H}(0)$ is added to the reaction term. After that, β, a learnable parameter, determines how much the source term is added. In other words, GREAD-ST learns how to utilize $\mathbf{H}(0)$ in the reaction term.

We compare GRAND++ (Thorpe et al., 2022) and GREAD-ST through experiments using the benchmark dataset in Table 22. GREAD-ST shows superiority in all datasets. In Table 23, we also show the performance by varying the number of layers for Cora. Our method is more robust to the oversmoothing problem than GRAND++.

Table 22. Comparison with GRAND++ and GREAD-ST

Model	Texas	Wisconsin	Cornell	Film	Squirrel	Chameleon	Cora	Citeseer	Pubmed
GRAND++	$77.57_{\pm5.00}$	$82.75_{\pm4.19}$	$81.89_{\pm5.28}$	$33.63_{\pm0.48}$	$40.06_{\pm1.70}$	$56.20_{\pm2.15}$	$88.15_{\pm1.22}$	$76.57_{\pm1.46}$	$88.50_{\pm0.35}$
GREAD-ST	$\mathbf{81.08}_{\pm\mathbf{5.67}}$	$\mathbf{86.67}_{\pm\mathbf{3.01}}$	$\mathbf{86.22}_{\pm\mathbf{5.98}}$	$\mathbf{37.66}_{\pm\mathbf{0.90}}$	$\mathbf{45.83}_{\pm\mathbf{1.40}}$	$\mathbf{63.03}_{\pm\mathbf{1.32}}$	$\mathbf{88.47}_{\pm\mathbf{1.19}}$	$\mathbf{77.25}_{\pm\mathbf{1.47}}$	$\mathbf{90.13}_{\pm\mathbf{0.36}}$

Table 23. Classification accuracy of GRAND++ and GREAD-ST with different depths on Cora dataset

Dataset	Model	Layer					
		2	4	8	16	32	64
Cora	GRAND++	$87.38_{\pm2.01}$	$88.15_{\pm1.22}$	$87.89_{\pm1.13}$	$87.73_{\pm0.96}$	$87.52_{\pm1.28}$	$87.73_{\pm1.30}$
	GREAD-ST	$\mathbf{88.17}_{\pm\mathbf{0.38}}$	$\mathbf{88.31}_{\pm\mathbf{0.39}}$	$\mathbf{88.39}_{\pm\mathbf{0.83}}$	$\mathbf{88.12}_{\pm\mathbf{0.53}}$	$\mathbf{88.47}_{\pm\mathbf{1.19}}$	$\mathbf{88.37}_{\pm\mathbf{0.98}}$

I. Well-posedness of GREAD

The well-posedness[1] of NODEs was already proved in Lyons et al. (2004, Theorem 1.3) under the mild condition of the Lipschitz continuity. Almost all activations, such as ReLU, Leaky ReLU, SoftPlus, Tanh, Sigmoid, ArcTan, and Softsign, have a Lipschitz constant of 1. Other common neural network layers, such as dropout, batch normalization, and other pooling methods, have explicit Lipschitz constant values. Therefore, the Lipschitz continuity of $\mathbf{f}$ can be fulfilled in some cases of GREAD, making the initial value problem in Eq. (7) a well-posed problem. However, some other functions are locally Lipschitz continuous. For instance, GREAD with the soft adjacency matrix does not satisfy the globally Lipschitz continuous property. Nevertheless, our experimental results show that GREAD can be properly trained and outperforms many baselines.

---

[1]A well-posed problem means i) its solution uniquely exists, and ii) its solution continuously changes as input data changes.

J. Statistical Testing on Cora Dataset

For proper statistical testing, we experiment more with 10 different seeds per split and perform statistical tests using a total of 100 experimental results. The unpaired t-test of GRAND and GREAD-BS is shown in Table 24. As shown, GREAD-BS has a clear improvement in all datasets compared to GRAND.

Table 24. Significance test between GRAND and GRAND-BS utilizing unpaired t-test

	Texas	Wisconsin	Cornell	Film	Squirrel	Chameleon	Cora	Citeseer	Pubmed
GRAND	$76.10_{\pm 6.03}$	$79.19_{\pm 5.26}$	$82.17_{\pm 5.99}$	$33.43_{\pm 1.29}$	$38.09_{\pm 1.38}$	$53.86_{\pm 2.04}$	$87.12_{\pm 1.74}$	$76.11_{\pm 1.24}$	$88.82_{\pm 0.50}$
GREAD-BS	$\mathbf{88.71_{\pm 3.24}}$	$\mathbf{89.10_{\pm 2.90}}$	$\mathbf{86.44_{\pm 7.03}}$	$\mathbf{39.90_{\pm 1.02}}$	$\mathbf{58.89_{\pm 1.11}}$	$\mathbf{70.04_{\pm 0.93}}$	$\mathbf{88.43_{\pm 0.59}}$	$\mathbf{77.48_{\pm 1.15}}$	$\mathbf{90.03_{\pm 0.49}}$
t-statistic	18.42	16.50	4.62	39.34	117.45	72.17	7.13	8.10	17.28
p-value	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05