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 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 17 baselines, our method, called GREAD, outperforms them in almost all cases. Further synthetic data experiments show that GREAD mitigates the oversmoothing and performs well for various homophily rates.

Introduction

Graphs are a useful data format that occurs frequently in real-world applications, e.g., computer vision and graphics, recommender systems, molecular chemistry inference, traffic forecasting, drug discovery, 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. 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.

| Model  | Diffusion | Reaction | Continuous |
|--------|-----------|----------|------------|
| GCN    | O         | X        | X          |
| GAT    | O         | X        | X          |
| FA-GCN | O         | Δ        | X          |
| GPR-GNN| O         | Δ        | X          |
| GRAND  | O         | X        | O          |
| BLEND  | O         | X        | O          |
| GREAD  | O         | O        | O          |

Table 1: A comparison table of existing methods. Our GREAD not only combines the diffusion and reaction processes, but also continuously interprets them. ‘Δ’ means that their designs correspond to a specific type of reaction.

Figure 1: An illustrative comparison between the diffusion equation in Eq. (9) (bottom) and our proposed blurring-sharpening reaction-diffusion equation in Eq. (8) (top) on a grid graph with one-dimensional node input features. The diffusion equation causes the problem of oversmoothing while the reaction-diffusion seeks a balance between smoothing and sharpening.

In Table 1 we compare recent methods. Most of them rely on diffusion processes while two of them (i.e., FA-GCN and GPR-GNN) partially utilize reaction processes (although the authors of the two methods were not aware of it). Those two...
methods, however, utilize limited forms of the reaction processes. This is because those methods were designed without considering reaction processes but by chance, they correspond to certain reaction processes. In this regard, there do not exist any methods that fully consider reaction processes.

To this end, we propose the concept of graph neural reaction-diffusion equation (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; Halatek and Frey 2018) (see Fig. 1). Since it is natural that nodes on a graph also constitute local clusters, we conjecture that reaction-diffusion equations are suitable for designing GNNs.

Our proposed model, GREAD, consists of three parts: an encoder, a reaction-diffusion layer, and an output layer (cf. Eqs. (4) to (9)). The reaction-diffusion layer has four different types in its core part: i) Fisher (F), ii) Allen-Cahn (AC), iii) Zeldovich (Z), and iv) Blurring-sharpening (BS). The first three reaction-diffusion equations are widely used in many natural science domains, e.g., biology, combustion, etc. In particular, the blurring-sharpening (BS) equation was designed by us for GNNs, which marks the best accuracy in many cases of our experiments.

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 17 baselines, which covers early to recent GNNs. Our proposed model, GREAD, consists of three parts: an encoder, a reaction-diffusion layer, and an output layer (cf. Eqs. (4) to (9)). The reaction-diffusion layer has four different types in its core part: i) Fisher (F), ii) Allen-Cahn (AC), iii) Zeldovich (Z), and iv) Blurring-sharpening (BS). The first three reaction-diffusion equations are widely used in many natural science domains, e.g., biology, combustion, etc. In particular, the blurring-sharpening (BS) equation was designed by us for GNNs, which marks the best accuracy in many cases of our experiments.

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 17 baselines, which covers early to recent GNNs. Our contributions can be summarized as follows:

1. We design a reaction-diffusion layer that incorporates i) three types of reaction equations popular in natural sciences and ii) one more type proposed by us.
2. We carefully integrate the four 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 17 baselines. Our method marks the best accuracy in almost all cases. The ranking and accuracy averaged over all the datasets are summarized in Table 2.

### Preliminaries & Related Work

#### Graph Neural Networks

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

#### Spatial and Spectral GNNs

GNNs have many variants and applications. We focus on a brief introduction of representation learning for nodes in supervised or semi-supervised classification tasks. Most existing approaches follow a message-passing framework and use a permutation-invariant local aggregation scheme to update each node’s representation, e.g., spatial GNNs. For example, GCN (Kipf and Welling 2017) averages features of each node’s neighbors, including the node’s self feature, to update its representation. GAT (Veličković et al. 2018) introduces an attention mechanism to learn aggregation weights over all neighbors. For fast and scalable GNN training, sampling-based methods have been developed, such as GraphSAGE (Hamilton, Ying, and Leskovec 2017) and FastGCN (Chen, Ma, and Xiao 2018). The simplifying approaches (Wu et al. 2019; Tiezzi et al. 2021; Zhu and Koniusz 2020) also make the GNN methods more efficient.

Recently, Balclard et al. (Balclard et al. 2021) bridge the gap between spectral and spatial GNNs and unify them into a single framework. ChebNet (Defferrard, Bresson, and Vandergheynst 2016) uses the Chebyshev polynomial to approximate filters. GraphHeat (Xu et al. 2019) uses the heat kernel to design graph filters. APPNP (Klicpera, Bojchevski, and Günnemann 2019) utilizes the personalized PageRank to set the filter weights. FA-GCN (Bo et al. 2021) can adaptively integrate different signals in the message passing process, and GPR-GNN (Chien et al. 2021) learns the polynomial filters via gradient descent on the polynomial coefficients.

### Table 2: The average ranking and accuracy of some selected high-performing models on 9 real-world datasets.

| Method   | Avg. Ranking | Avg. Accuracy |
|----------|--------------|---------------|
| GREAD-BS | 1.44         | 74.94\textsuperscript{*} |
| GREAD-AC | 3.94         | 73.55\textsuperscript{†} |
| GREAD-Z  | 4.06         | 73.26          |
| GREAD-F  | 4.28         | 73.73          |
| GCNI     | 7.00         | 70.16          |
| BLENDE   | 7.22         | 71.79          |
| H2GCN    | 7.50         | 71.33          |
| FA-GCN   | 11.00        | 69.86          |

The table shows the average ranking and accuracy for different methods on 9 real-world datasets. The best performance is indicated by \textsuperscript{*} (resp. \textsuperscript{†}) indicating an improvement over GCNI (resp. BLENDE) is statistically significant ($p < 0.05$) under the Wilcoxon signed-rank test.
where one of the most advanced solvers. Iterations is not deterministic. DOPRI is widely regarded as a particular method for ODEs. It is an adaptive solver that dynamically adjusts the step size based on estimated potential errors. As a result, in the case of DOPRI, the number of iterations is not deterministic. DOPRI is widely regarded as one of the most advanced solvers.

Neural Ordinary Differential Equations (NODEs)

Neural ordinary differential equations (NODEs) solve the initial value problem (IVP), which involves a Riemann integral problem, from the explicit dynamic approach. NODEs are a continuous generalization of residual networks.

\[ h(t_{i+1}) = h(t_i) + \int_{t_i}^{t_{i+1}} f(h(t), t; \theta_f) dt, \]

where \( \theta_f \) approximates the time-derivative of \( h \), i.e., \( \dot{h} = \frac{dh(t)}{dt} \). 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 and Prince 1980). The Euler method is written as follows:

\[ h(t + h) = h(t) + \mathbf{r} \cdot \mathbf{f}(h(t)), \]

where \( \mathbf{r} \) is a reaction term, and \( \mathbf{r} \) is in the reaction-diffusion form. \( \mathbf{f}(h(t)) \) is a reaction term, and \( \theta_f \) is a trainable parameter.

Other ODE solvers use more complicated methods to update \( h(t + \tau) \) from \( h(t) \). For instance, the fourth-order Runge–Kutta (RK4) method uses the following method:

\[ h(t + \tau) = h(t) + \frac{\tau}{6} (f_1 + 2f_2 + 2f_3 + f_4), \]

where \( f_1 = f(h(t)), f_2 = f(h(t) + \tau f_1), f_3 = f(h(t) + \tau f_3), \) and \( f_4 = f(h(t) + 2\tau f_3) \).

In order to solve the above integral problem, theoretically, we need to iterate one of the fixed-step ODE solvers \( T/\tau \) times since each iteration updates \( h(t) \) to \( h(t + \tau) \). The DOPRI method, on the other hand, is an adaptive solver that dynamically adjusts the step size based on estimated potential errors. As a result, in the case of DOPRI, the number of iterations is not deterministic. DOPRI is widely regarded as one of the most advanced solvers.

Proposed Method

After describing an overview of our method, we describe its detailed designs, followed by its training algorithm.

Overview of GREAD

Given a graph \( G \) with its node feature matrix \( X \) and its symmetric normalized Laplacian (resp. symmetric normalized adjacency) matrix \( L \) (resp. \( A \)), GREAD can be written as follows:

\[ H(0) = e(X), \]

\[ H(T) = H(0) + \int_0^T f(H(t)) dt, \]

\[ \hat{y} = o(H(T)), \]

where \( f(H(t)) = -\alpha LH(t) + \beta r(H(t), A) \) is in the reaction-diffusion form. \( r(H) \) is a reaction term, and \( \alpha \) and \( \beta \) are trainable parameters.

The encoder \( e \) has a couple of fully-connected layers with rectified linear unit (ReLU) activations. The output layer \( o \) 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 \( r \), 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.

Soft Adjacency Matrix Generation

Given a graph \( G \), one can use its original symmetric normalized adjacency matrix \( A \in [0, 1]^{[V \times V]} \) for our method. However, we also provide the method to generate a soft adjacency matrix, in which case \( A \in [0, 1]^{[V \times V]} \). For our experiments, we test both of them.

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

\[ A_{[i,j]} = \text{softmax} \left( \frac{(W_K H_i)^T W_Q H_j}{d_K} \right), \]

where \( A_{[i,j]} \) means the \((i,j)\)-th element of \( A \), \( W_K \) and \( W_Q \) are trainable parameters, and \( d_K \) is the scale factor. \( H_i, H_j \) are trainable embedding vectors of nodes \( i, j \).
Reaction-diffusion Layer

Eq. (5) is the main processing layer, called the reaction-diffusion layer, in our method. Given the definition of $f$, $-LH(t)$ is a diffusion term, which 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; Chen et al. 2021a,b). It is known that the diffusion terms cause the problem of oversmoothing, which means that the last hidden states of nodes become too similar when applying only the diffusion processing too much (without the reaction processing). To this end, many models prefer shallow architectures that do not cause the oversmoothing problem (Wu et al. 2019; Kipf and Welling 2017) or use heuristic methods to prevent it (Zhao and Akoglu 2020; Chen, Ma, and Xiao 2018; Chen et al. 2020; Li et al. 2019; Liu, Gao, and Ji 2020; Huang et al. 2018; Chen, Zhu, and Song 2018).

In our case, we prevent the oversmoothing problem by adding the reaction term $r$ and solve Eq. (5) with ODE solvers (Dormand and Prince 1980) — in other words, our reaction-diffusion layer is continuous, which is yet another distinguishing point in our method since many other models are based on discrete layers (Kipf and Welling 2017; Bo et al. 2021; Chien et al. 2021; Zhu et al. 2020). We consider the following options for $r$:

$$
\begin{align*}
  r(H(t), A) &:= \begin{cases} 
  H(t) \odot (1 - H(t)), & \text{if Fisher (F)} \\
  H(t) \odot (1 - H(t)^2), & \text{if Allen-Cahn (AC)} \\
  (A - A^2)H(t), & \text{if Blurring-Sharpening (BS)} \\
  (A - A^2)H(t) - (H(t) - H(t)^2), & \text{if Zeldovich (Z)}
  \end{cases} \\
\end{align*}
$$

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 and Cahn 1979). Z is a generalized equation that describes the phenomena that occur in combustion theory (Gilding and Kersner 2004). The last BS is specially designed by us for GNNs, which we will describe shortly.

Blurring-Sharpening Process Our proposed blurring-sharpening process is to alternately perform the blurring and the sharpening operations in a 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 blurring. We also use the same blurring operation at first:

$$
\begin{align*}
  B(t + h) &= H(t) - LH(t), \\
  &\Rightarrow H(t) + (A - I)H(t), \\
  &\Rightarrow AH(t).
\end{align*}
$$

We then proposed to apply the following high-pass graph convolutional filtering or sharpening process to $B(t + h)$.

Algorithm 1: How to train our proposed GREAD

| Input: Training data $D_{train}$, Validating data $D_{val}$, Maximum iteration number $max\_iter$ |
| 1. Initialize model parameters $\theta$; |
| 2. $k \leftarrow 0$; |
| 3. while $k < max\_iter$ do |
| 4. Construct a mini-batch $B$ from $D_{train}$; |
| 5. Train $\theta$ with Eq. (13) and $B$; |
| 6. Validate and update the best parameters, $\theta^*$ with $D_{val}$; |
| 7. $k \leftarrow k + 1$; |
| 8. return $\theta^*$; |

In other words, there is a sharpening process following the above blurring process in a layer:

$$
\begin{align*}
  H(t + h) &= B(t + h) + LH(B(t + h)), \\
  &\Rightarrow AH(t) + L(AH(t)), \\
  &\Rightarrow 2AH(t) - A^2H(t), \\
  &\Rightarrow (I - A^2)AH(t), \\
  &\Rightarrow (I + L)(I - L)H(t), \\
  &\Rightarrow H(t) - LH^2H(t), \\
  &\Rightarrow H(t) - (I - A)^2H(t), \\
  &\Rightarrow H(t) - (I - A)H(t) + (A - A^2)H(t), \\
  &\Rightarrow H(t) - LH(t) + (A - A^2)H(t).
\end{align*}
$$

Therefore, we can derive the following difference equation:

$$
\begin{align*}
  H(t + h) - H(t) &= -LH(t) + (A - A^2)H(t). \\
  \frac{dH(t)}{dt} &= -LH(t) + (A - A^2)H(t),
\end{align*}
$$

which is a reaction-diffusion equation where $r(H(t), A) := (A - A^2)H(t)$.

Training Algorithm

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

$$
\mathcal{L} = \sum_{i}^{n} y_i^T \log \hat{y}_i,
$$

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

Well-posedness of Training The well-posedness\(^2\) of NODEs was already proved in (Lyons, Caruana, and Levy 2004, Theorem 1.3) under the mild condition of the Lipschitz continuity. We show that training our NODE layers is also a well-posed problem. Almost all activations, such

\(^2\)A well-posed problem means i) its solution uniquely exists, and ii) its solution continuously changes as input data changes.
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 $f$, $m$ and $j$, for all $i$ can be fulfilled in our case, making it a well-posed training problem. Our training algorithm solves a well-posed problem so its training process is stable in practice.

### 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. 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, TORCHDIFEQ 0.2.3, CUDA 11.3, NVIDIA Driver 465.19, i9 CPU, and NVIDIA RTX 3090.

### Node Classification on Real-world Datasets

#### Real-world Datasets

We evaluate the performance of GREAD and existing GNNs on a variety of real-world datasets. We consider 6 heterogeneous datasets with low homophily ratios used in (Pei et al. 2020): (ii) Chameleon, Squirrel (Rozemberczki, Allen, and Sarkar 2021), (iii) Film (Tang et al. 2009), iv, v, vi) Texas, Wisconsin and Cornell from WebKB[^1]. 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, Cohen, and Salakhutdinov 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.

#### 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, Bresson, and Vandergheynst 2016), GCN (Kipf and Welling 2017), GAT (Velickovic et al. 2018), GraphSAGE (Hamilton, Ying, and Leskovec 2017), and SGC (Wu et al. 2019).

2. The next group includes the GNN methods designed for heterophilic settings: Geom-GCN (Pei et al. 2020), H2GCN (Zhu et al. 2020), FA-GCN (Bo et al. 2021), and GPR-GNN (Chien et al. 2021).

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

4. The last group contains continuous-time GNN methods: GDE (Poli et al. 2019), CGNN (Xhonneux, Qu, and Tang 2020), GRAND (Chamberlain et al. 2021b), and BLEND (Chamberlain et al. 2021a).

#### Hyperparameters

For our method, we test with the following hyperparameter configurations: we train for 200 epochs using the Adam optimizer. The dimensionality of $H$, denoted $\text{dim}(H)$, is in $\{32, 64, 128, 256\}$. $T$ in Eq. (5) is from 1.0 to 4.0. The learning rate is set to from $1 \times 10^{-4}$ to $2 \times 10^{-2}$. The weight decay is 0 to 0.01. The detailed search space and other hyperparameters are in Appendix. We also list the best hyperparameter configuration for each data in Appendix. If a baseline’s accuracy is known and its 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 of all real-world datasets. GREAD-BS is ranked at the top with the average ranking of 1.44, followed by GREAD-AC and GREAD-Z. The fourth-ranked method, GREAD-F, shows a clearly higher ranking in comparison with GCNII and others. In Fig. 2, we visualize the hidden node features at each ODE time step of Eq. (5), and the reaction-diffusion processes of GREAD lead to local clusters after several steps.

Tables 4 presents the detailed classification performance. As reported, our method marks the best accuracy in all cases except for Squirrel and Citeseer. GCNII and H2GCN show comparable accuracy values from time to time. However,

### Table 3: Benchmark dataset properties and statistics

| Dataset       | Texas | Wisconsin | Cornell | Film | Squirrel | Chameleon | Cora | Citeseer | PubMed |
|---------------|-------|-----------|---------|------|----------|-----------|------|----------|--------|
| Classes       | 5     | 5         | 5       | 5    | 5        | 6         | 7    | 7        | 3      |
| Features      | 1,703 | 1,703     | 1,703   | 932  | 2,089    | 235       | 1,433| 3,703    | 500    |
| Nodes         | 185   | 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   |

[^1]: http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/webkb

---

*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.*
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. BLEND fails to mark the best or the second-best place in all cases. While GREAD-BS is the best method overall, GREAD-F is the best method for Texas and is the second-best for Chameleon. GREAD-AC marks the best accuracy on Cornell and third place in Film and Chameleon.

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 improves the model accuracy in Cornell except for GREAD-F.

Next, we also perform the ablation study on $\beta$. $\beta$ can be either a scalar parameter (denoted as SC) or a learnable vector parameter (denoted as VC). We compare them in Table 6.
The Dirichlet Energy

We can define the oversmoothing via the evolution of the Dirichlet energy. The Dirichlet energy $E$ of an undirected graph $G$ is defined as follows:

$$E(H, A) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \sum_{j \in N_i} A[i,j] ||H_i - H_j||^2,$$  \hspace{1cm} (14)

where $H_i, H_j$ mean $i$-th and $j$-th rows, respectively.

The oversmoothing phenomenon means that as the depth increase, all node features converge to a constant. Thus, $E(H, A)$ decays to zero asymptotically in time. We will show that our proposed method allows mitigating oversmoothing via the evolution of the Dirichlet energy.

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.

Different Homophily Levels

Experimental Environments 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, 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.

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 one special reaction term, called Blurring-sharpening (BS), designed by us for GNNs. Therefore, our reaction-diffusion layer has four types. We consider a comprehensive set of 9 real-world datasets with various homophily difficulties and 17 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.
References
Allen, S. M.; and Cahn, J. W. 1979. A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening. *Acta metallurgica*, 27(6): 1085–1095.

Balcilar, M.; Renton, G.; Héroux, P.; Gaùzèrè, B.; Adam, S.; and Honeine, P. 2021. Analyzing the Expressive Power of Graph Neural Networks in a Spectral Perspective. In *ICLR*.

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

Biewald, L. 2020. Experiment Tracking with Weights and Biases. Software available from wandb.com.

Bo, D.; Wang, X.; Shi, C.; and Shen, H. 2021. Beyond Low-frequency Information in Graph Convolutional Networks. In *AAAI*.

Chamberlain, B. P.; Rowbottom, J.; Eynard, D.; Di Giovanni, F.; Xiaowen, D.; and Bronstein, M. M. 2021a. Beltrami Flow and Neural Diffusion on Graphs. In *NeurIPS*.

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

Chen, J.; Ma, T.; and Xiao, C. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR*.

Chen, J.; Zhu, J.; and Song, L. 2018. Stochastic Training of Graph Convolutional Networks with Variance Reduction. In *ICML*.

Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; and Li, Y. 2020. Simple and Deep Graph Convolutional Networks. In *ICML*.

Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural Ordinary Differential Equations. In *NeurIPS*.

Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*.

Choi, J.; Choi, H.; Hwang, J.; and Park, N. 2022. Graph Neural Controlled Differential Equations for Traffic Forecasting. In *AAAI*.

Choi, J.; Jeon, J.; and Park, N. 2021. LT-OCF: Learnable-Time ODE-based Collaborative Filtering. In *CIKM*.

Coifman, R. R.; Lafon, S.; Lee, A. B.; Maggioni, M.; Nadler, B.; Warner, F.; and Zucker, S. W. 2005. 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.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NeurIPS*.

Deshpande, Y.; Sen, S.; Montanari, A.; and Mossel, E. 2018. Contextual Stochastic Block Models. In *NeurIPS*.

Desquesnes, X.; Elmoataz, A.; and Lézoray, O. 2013. 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.

Dormand, J.; and Prince, P. 1980. A family of embedded Runge-Kutta formulæ. *Journal of Computational and Applied Mathematics*, 6(1): 19–26.

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

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

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

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

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

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

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

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*.

Halatek, J.; and Frey, E. 2018. Rethinking pattern formation in reaction–diffusion systems. *Nature Physics*, 14(5): 507–514.

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

Huang, W.; Zhang, T.; Rong, Y.; and Huang, J. 2018. Adaptive Sampling Towards Fast Graph Representation Learning. In *NeurIPS*.

Hwang, J.; Choi, J.; Choi, H.; Lee, K.; Lee, D.; and Park, N. 2021. Climate Modeling with Neural Diffusion Equations. In *ICDM*, 230–239.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Klicpera, J.; Bojchevski, A.; and Günnemann, S. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.

Li, G.; Muller, M.; Thabet, A.; and Ghanem, B. 2019. DeepGCNs: Can GCNs Go As Deep As CNNs? In *ICCV*.

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

Liu, M.; Gao, H.; and Ji, S. 2020. Towards Deeper Graph Neural Networks. In *KDD*, 338–348.

Lyons, T.; Caruna, M.; and Lévy, T. 2004. *Differential Equations Driven by Rough Paths*. Springer. École D’Eté de Probabilités de Saint-Flour XXXIV - 2004.
McCallum, A. K.; Nigam, K.; Rennie, J.; and Seymore, K. 2000. Automating the construction of internet portals with machine learning. Information Retrieval, 3(2): 127–163.

Monti, F.; Boscaini, D.; Masci, J.; Rodolà, E.; Svoboda, J.; and Bronstein, M. M. 2017. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In CVPR, 5425–5434.

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

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

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

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

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

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

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

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

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

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

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

Wang, Y.; Wang, Y.; Yang, J.; and Lin, Z. 2021. Dissecting the Diffusion Process in Linear Graph Convolutional Networks. In NeurIPS.

Wu, F.; Zhang, T.; de Souza, A. H.; Fifty, C.; Yu, T.; and Weinberger, K. Q. 2019. Simplifying Graph Convolutional Networks. In ICML.

Xhonneux, L.-P. A. C.; Qu, M.; and Tang, J. 2020. Continuous Graph Neural Networks. In ICML.

Xu, B.; Shen, H.; Cao, Q.; Cen, K.; and Cheng, X. 2019. Graph Convolutional Networks using Heat Kernel for Semi-supervised Learning. In IJCAI.

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

Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In ICML.
The Full Ranking of Table 2

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

| Method       | Avg. Ranking | Avg. Accuracy |
|--------------|--------------|---------------|
| GREAD-BS     | 1.44         | 74.94         |
| GREAD-AC     | 3.94         | 73.55         |
| GREAD-Z      | 4.06         | 73.26         |
| GREAD-F      | 4.28         | 73.73         |
| GCNII        | 7.00         | 70.16         |
| BLEND        | 7.22         | 71.79         |
| H2GCN        | 7.50         | 71.33         |
| FA-GCN       | 11.00        | 69.86         |
| GRAND        | 11.33        | 68.94         |
| GraphSAGE    | 11.94        | 69.50         |
| GPR-GNN      | 11.94        | 67.45         |
| GDE          | 12.89        | 67.40         |
| Geom-GCN     | 13.17        | 63.87         |
| GCN          | 13.39        | 62.77         |
| MLP          | 13.78        | 66.26         |
| CGNN         | 14.00        | 63.96         |
| ChebNet      | 14.67        | 67.98         |
| GAT          | 15.56        | 60.23         |
| PairNorm     | 16.00        | 61.68         |
| JKNNet       | 17.00        | 60.77         |
| SGC          | 18.89        | 58.34         |

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

Computational Complexity

The space complexity of GREAD is dominated by evaluating the soft adjacency matrix in Eq. (7), which is $O(|\mathcal{E}| \text{dim}(\mathbf{H}))$, 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. (5). Our proposed model has different complexity depending on the reaction term $r$ in Eq. (8).

If we set the adjacency matrix and $\beta$ to OA and SC respectively, the time complexity of the one-step GREAD-BS computation becomes $O(n_{\tau}(|\mathcal{E}| + |\mathcal{E}_2|) \text{dim}(\mathbf{H}) + |\mathcal{E}| d_{\text{max}})$, where $n_{\tau}$, $|\mathcal{V}|$ and $d_{\text{max}}$ 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 $O(|\mathcal{E}| d_{\text{max}})$ because $d_{\text{max}}$ is equal to the maximum number of non-zeros in any row of $\mathbf{A}$. The sparse matrix multiplication of $\mathbf{A}^2 \mathbf{H}(t)$ takes $O(|\mathcal{E}_2| d_{\text{max}})$, 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 $O(n_{\tau}(|\mathcal{E}| + \text{dim}(\mathbf{H})^k) \text{dim}(\mathbf{H}))$, where $k = 1$. In the case of GREAD-AC and GREAD-Z, their $k$ values are 2 and 3, respectively.

Additional Details for Experiments

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:

$$\text{Homophily ratio} = \frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} \frac{\sum_{v \in \mathcal{N}_u} \left( y_u = y_v \right)}{|\mathcal{N}_v|}.$$  \hspace{1cm} (15)

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:

1. Chameleon and Squirrel are subgraphs of web pages in Wikipedia (Rozemberczki, Allen, and Sarkar, 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, Cohen, and Salakhutdinov 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 8, we summarize the properties of the synthetic Cora networks we used.

| 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        |

Table 8: 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).

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.

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 hyperparameter search but list the full hyperparameter list we used in Table 9.

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 10, we list the hyperparameter range we consider.

Node Classification on Real-world Datasets  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 11. 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 12 to 15.
| Hyperparameters   | Value  |
|-------------------|--------|
| epochs            | 100    |
| adjacency matrix  | OA     |
| $\alpha$         | SC     |
| $\beta$          | VC     |
| learning rate     | 0.001  |
| weight decay      | $5 \times 10^{-4}$ |
| dropout           | 0.0    |
| input dropout     | 0.5    |
| dim($\mathbf{H}$) | 2      |
| step size $\tau$ | 1.0    |
| time $T$          | 40     |
| ODE solver        | Euler  |

Table 9: Hyperparameter for the cSBM synthetic network

| Hyperparameters   | Search Space                           |
|-------------------|----------------------------------------|
| epochs            | 100                                    |
| adjacency matrix  | \{OA, SA\}                            |
| $\alpha$         | \{SC, VC\}                            |
| $\beta$          | \{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 $\tau$ | \{0.1, 0.5, 1.0\}                     |
| time $T$          | \{1, 2, 3, 4\}                         |
| ODE solver        | Euler                                  |

Table 10: Hyperparameter search space for the synthetic Cora network

| Hyperparameters   | Search Space                                      |
|-------------------|---------------------------------------------------|
| epochs            | 200                                               |
| adjacency matrix  | \{OA, SA\}                                        |
| $\alpha$         | \{SC, VC\}                                        |
| $\beta$          | \{SC, VC\}                                        |
| learning rate     | $[1 \times 10^{-4}, 2 \times 10^{-2}]$            |
| weight decay      | $[0, 1 \times 10^{-2}]$                           |
| dropout           | $[0, 0.6]$                                        |
| input dropout     | $[0, 0.6]$                                        |
| dim($\mathbf{H}$) | \{32, 64, 128, 256\}                             |
| step size $\tau$ | $[0.1, 1.0]$                                      |
| time $T$          | $[0.1, 4.0]$                                      |
| ODE solver        | \{Euler, RK4, DOPRI5\}                            |

Table 11: Hyperparameter search space for real-world datasets
| Hyperparameters | Texas | Wisconsin | Cornell | Film | Squirrel | Chameleon | Cora | Citeseer | PubMed |
|-----------------|-------|-----------|---------|------|----------|-----------|------|----------|--------|
| adjacency matrix| OA    | SA        | SA      | SA   | SA       | SA        | SA   | SA       | SA     |
| \( \alpha \)    | SC    | SC        | SC      | SC   | SC       | SC        | SC   | SC       | SC     |
| \( \beta \)     | VC    | VC        | VC      | VC   | VC       | VC        | SC   | SC       | SC     |
| learning rate    | 0.0100| 0.0154    | 0.0084  | 0.0045| 0.0137   | 0.0120    | 0.0105| 0.0024   | 0.0108 |
| weight decay     | 0.0175| 0.0041    | 0.3109  | 0.0094| 0.0006   | 0.0007    | 0.0060| 0.0146   | 0.0005 |
| input dropout    | 0.49  | 0.54      | 0.49    | 0.01 | 0.49     | 0.55      | 0.53  | 0.50     | 0.36   |
| dropout          | 0.39  | 0.48      | 0.29    | 0.04 | 0.05     | 0.22      | 0.45  | 0.47     | 0.26   |
| \( \text{dim}(\mathbf{H}) \) | 256   | 128       | 128    | 64   | 128      | 64        | 128   | 64       | 64     |
| step size \( \tau \) | 0.4   | N/A       | 0.18   | 1.0  | 1.0      | 0.25      | 0.5   | 0.8      |        |
| time \( T \)    | 1.15  | 0.4       | 0.18   | 0.14 | 3.96     | 2.92      | 3.49  | 2.35     | 1.74   |
| ODE solver       | Euler | DOPRI5    | Euler   | RK4  | Euler    | RK4       | RK4  | RK4      | RK4    |

Table 12: Best hyperparameters of GREAD-BS

| Hyperparameters | Texas | Wisconsin | Cornell | Film | Squirrel | Chameleon | Cora | Citeseer | PubMed |
|-----------------|-------|-----------|---------|------|----------|-----------|------|----------|--------|
| adjacency matrix| SA    | SA        | SA      | SA   | SA       | SA        | SA   | SA       | SA     |
| \( \alpha \)    | VC    | SC        | VC      | VC   | SC       | SC        | SC   | SC       | SC     |
| \( \beta \)     | 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   |
| \( \text{dim}(\mathbf{H}) \) | 256   | 128       | 128    | 128  | 128      | 32        | 256   | 128      |        |
| step size \( \tau \) | 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       | RK4  | RK4      | RK4    |

Table 13: 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     |
| \( \alpha \)    | VC    | VC        | SC      | SC   | SC       | SC        | SC   | SC       | SC     |
| \( \beta \)     | VC    | VC        | VC      | VC   | VC       | VC        | SC   | VC       | VC     |
| learning rate    | 0.0136| 0.0081    | 0.0311  | 0.0001| 0.0020   | 0.0007    | 0.0469| 0.0140   | 0.0006 |
| weight decay     | 0.0070| 0.0083    | 0.0084  | 0.0027| 0.0025   | 0.0038    | 0.0039| 0.0029   | 0.0124 |
| input dropout    | 0.40  | 0.45      | 0.49    | 0.46 | 0.46     | 0.52      | 0.52  | 0.40     | 0.47   |
| dropout          | 0.30  | 0.20      | 0.29    | 0.48 | 0.28     | 0.35      | 0.40  | 0.49     | 0.26   |
| \( \text{dim}(\mathbf{H}) \) | 256   | 128       | 128    | 128  | 128      | 256       | 128   | 64       | 128    |
| step size \( \tau \) | 1.0   | 0.5       | 0.75    | 1.0  | 1.0      | 1.0       | 0.1   | 0.9      | 1      |
| 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 | RK4     | Euler     | RK4  | RK4      | RK4    |

Table 14: Best hyperparameters of GREAD-AC

| Hyperparameters | Texas | Wisconsin | Cornell | Film | Squirrel | Chameleon | Cora | Citeseer | PubMed |
|-----------------|-------|-----------|---------|------|----------|-----------|------|----------|--------|
| adjacency matrix| OA    | SA        | SA      | SA   | SA       | SA        | SA   | SA       | SA     |
| \( \alpha \)    | VC    | SC        | SC      | SC   | VC       | VC        | SC   | VC       | VC     |
| \( \beta \)     | SC    | SC        | SC      | VC   | SC       | VC        | VC   | SC       | 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   |
| \( \text{dim}(\mathbf{H}) \) | 256   | 128       | 256    | 64   | 128      | 256       | 64   | 64       | 64     |
| step size \( \tau \) | 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 15: Best hyperparameters of GREAD-Z
Training Time

We present the training time of GREAD and some selected baselines in Fig. 7. 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. The ODE solvers significantly affect training time. For example, using RK4 increases the time by 76.4% more than Euler solver.

![Figure 7: Average running time per epoch (ms) on Cora dataset when T = 3, \( \tau = 1.0 \), SC, and OA.](image)

Ablation Studies

Tables 16 to 17 show the results of our additional ablation studies in the remaining datasets that are not reported in our main paper. In Table 16, 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 17, we compare two types of \( \beta \). \( \beta \) can be either a scalar parameter (SC) or a learnable vector parameter (VC). In almost cases, it shows better performance when the type of \( \beta \) is VC.

| Dataset | A   | GREAD-BS | GREAD-F | GREAD-AC | GREAD-Z |
|---------|-----|----------|---------|----------|---------|
| Texas   | OA  | 87.57 ± 5.70 | 86.49 ± 5.69 | 85.41 ± 5.16 | 87.30 ± 4.20 |
|         | SA  | 85.41 ± 2.76 | 88.11 ± 5.01 | 85.95 ± 2.65 | 86.49 ± 3.20 |
| Wisconsin | OA | 87.45 ± 3.53 | 86.47 ± 4.16 | 85.49 ± 3.94 | 86.28 ± 3.62 |
|         | SA  | 88.04 ± 3.77 | 86.47 ± 4.84 | 85.69 ± 5.04 | 85.29 ± 4.32 |
| Squirrel | OA | 47.03 ± 1.31 | 37.85 ± 1.11 | 38.07 ± 1.71 | 38.43 ± 1.37 |
|         | SA  | 51.07 ± 1.47 | 46.16 ± 1.44 | 45.10 ± 2.11 | 46.25 ± 1.72 |
| Chameleon | OA | 61.21 ± 1.78 | 59.80 ± 1.54 | 58.93 ± 1.92 | 54.45 ± 2.29 |
|         | SA  | 67.98 ± 1.17 | 65.20 ± 1.65 | 65.09 ± 1.08 | 62.22 ± 1.44 |
| Cora    | OA  | 87.34 ± 1.34 | 86.72 ± 1.17 | 86.88 ± 1.09 | 86.90 ± 1.02 |
|         | SA  | 88.39 ± 0.87 | 88.07 ± 0.96 | 88.09 ± 1.35 | 88.13 ± 0.88 |
| Citeseer | OA | 77.18 ± 1.45 | 76.29 ± 1.74 | 76.73 ± 1.52 | 76.69 ± 1.97 |
|         | SA  | 77.53 ± 1.89 | 77.04 ± 1.86 | 77.31 ± 1.80 | 77.39 ± 1.73 |
| Pubmed  | OA  | 89.98 ± 0.38 | 87.99 ± 0.41 | 88.31 ± 0.44 | 88.83 ± 0.37 |
|         | SA  | 90.21 ± 0.47 | 88.64 ± 0.45 | 89.82 ± 0.38 | 89.08 ± 0.46 |

Table 16: Ablation study on soft adjacency matrix
| Dataset  | $\beta$ (SC) | $\alpha_1$ (VC) | $\alpha_2$ (SC) | $\alpha_2$ (VC) | $\alpha_3$ (SC) | $\alpha_3$ (VC) | $\alpha_4$ (SC) | $\alpha_4$ (VC) |
|---------|--------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Cornell | 85.14 ± 5.57 | 85.41 ± 6.75    | 85.41 ± 6.96    | 84.60 ± 6.17    | 85.41 ± 6.42    | 87.03 ± 4.95    | 83.24 ± 5.14    | 86.22 ± 6.45    |
| Film    | 37.09 ± 1.15 | 36.53 ± 1.04    | 37.21 ± 1.10    | 37.01 ± 1.11    | 37.49 ± 0.48    | 36.76 ± 0.99    | 36.70 ± 0.69    | 37.14 ± 0.63    |
| Squirrel| 42.74 ± 1.34 | 44.88 ± 1.62    | 39.61 ± 1.69    | 40.33 ± 2.06    | 51.07 ± 1.47    | 46.16 ± 1.44    | 46.25 ± 1.72    | 46.80 ± 1.49    |
| Chameleon| 62.02 ± 1.86 | 61.80 ± 1.80    | 56.56 ± 2.28    | 59.17 ± 1.26    | 67.98 ± 1.17    | 65.09 ± 1.08    | 61.93 ± 1.03    | 66.13 ± 1.44    |
| Cora    | 87.45 ± 1.08 | 88.07 ± 0.96    | 88.01 ± 0.85    | 88.13 ± 0.40    | 88.39 ± 0.87    | 88.03 ± 1.01    | 88.13 ± 0.88    | 88.39 ± 0.87    |
| Citeeex | 77.33 ± 1.74 | 76.70 ± 1.75    | 75.83 ± 1.36    | 76.83 ± 1.16    | 77.53 ± 1.89    | 77.01 ± 1.86    | 77.31 ± 1.80    | 77.39 ± 1.73    |
| Pubmed  | 89.96 ± 0.42 | 87.51 ± 0.44    | 88.31 ± 0.44    | 88.83 ± 0.37    | 90.21 ± 0.47    | 90.09 ± 0.31    | 90.10 ± 0.36    | 90.11 ± 0.27    |

Table 17: Ablation study on $\beta$

**Sensitivity Analyses**

In Figs. 8 and 9 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 8: Sensitivity to $T$
Figure 9: Sensitivity to step size
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.

Figure 10: Visualization of networks
Additional Experimental Results on Synthetic Datasets

Ablation Studies on $\beta$

We perform the ablation study on $\beta$ from the perspective of the Dirichlet energy. $\beta$ 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-BS, and GREAD-AC, VC conserves more energy than SC, so the reaction term multiplied with $\beta$ successfully mitigates the oversmoothing problem.

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.