Unleashing the Power of High-pass Filtering in Continuous Graph Neural Networks

Acong Zhang zac1328682511@gmail.com and **Ping Li** dping.li@gmail.com
School of Computer Science, Southwest Petroleum University

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

Recent Continuous Graph Neural Networks (CGNNs) have attracted great attention due to its merits of infinite depth without oversmoothing. However, most of the existing CGNNs perform low-pass filtering in nature, as they are derived from discrete Laplacian-smoothing based graph neural networks (GNNs). While prior research has shown the promising results of high-pass filtering for node representation learning, particularly on heterophilous graphs, there remains a need to extend it to continuous domain and explore the synergy between two filtering channels. In this paper, by leveraging low-pass and high-pass filtering, we propose a novel dual-channel continuous graph neural network architecture to address this gap. In particular, we introduce a dimension masking method to coordinate the contribution of all low and high pass filtered feature dimensions to node classification. Our aim is to deepen the understanding of the link between high and low filters, unraveling their distinct roles in learning node representations. To evaluate the effectiveness of our framework, we conduct extensive experiments focusing on the node classification task of heterophilous graphs. Our results demonstrate the competitive performance of our approach, showcasing its robustness to oversmoothing.

Keywords: Continuous graph neural network, high-pass filtering, semi-supervised learning, node classification.

1. Introduction

In recent years, graph neural networks(GNNs) (Bruna et al., 2014; Defferrard et al., 2016; Hamilton et al., 2017; Xu et al., 2019; Huang et al., 2023a) have become one of the most important techniques to solve a diversity of machine learning problems on non-Euclidean data (Tu and Neumann, 2022; Bastos et al., 2021; Jin et al., 2020) and achieved promising performance, such as nodes classification (Wang et al., 2020), link prediction (Li et al., 2019), chemical property prediction (Gilmer et al., 2017), and so on and so forth.

Along with the developments of more advanced GNNs, thoroughly understanding of both of its power and pitfalls has drawn great attention. As well-know limitation of GNNs is over-smoothing, which refers to the exponentially performance degradation for increasing number of layers. In an attempt to better resolve this issue, many efforts have been made to understand GNNs from different perspectives (Bodnar et al., 2022; Kang et al., 2021; Song et al., 2021), among which continuous graph neural networks (CGNNs) have become a rapidly emerging direction for mitigating over-smoothing. Different from most of the existing discrete message-passing based GNNs (Chen et al., 2020), CGNNs are built on various continuous diffusion dynamics (Xhonneux et al., 2020; Chamberlain et al., 2021b;

Song et al., 2022), so that the convolution can be performed with infinite number of layers (i.e., continuous layer) without loss of performance. Nevertheless, most of modern CGNNs are based on the homophily assumption, that is, connected nodes are similar, which limits the generality of CGNNs in real-world applications. We observe that the root of this limitation lies in the way of feature smoothing. In fact, most CGNNs perform Laplacian low-pass filtering (Xhonneux et al., 2020), which tend to preserves the similarity between neighboring nodes, while discarding the distinguishing features between them. However, the latter is decisive for node classification on heterophilious graphs.

In this paper, we particularly focus on improving performance of CGNNs on heterophilious graphs from the viewpoint of the interplay between low and high pass filtering. In the field of discrete GNNs, it is well known that low-pass filtering based models perform well on homophilous graphs (where neighboring nodes are more likely to be similar to each other in features or classification) (Nt and Maehara, 2019), while in contrast high-pass filtering based models are better at dealing with heterophilious graphs (Zhu et al., 2020). In this regard, models that can be both low and high pass filtering are preferable (Huang et al., 2023b; Yan et al., 2022). However, most of the existing models adopt sum or weighted sum aggregation of the features from low and high pass filters. This manner somehow limits the expressiveness power of these two pass filters in two ways: *First*, the current feature aggregation requires dimension alignment, i.e., features from two filtering channels need to have the same dimensionality, which can impair the flexibility of feature learning. *Second*, the summation operation may compromise the contribution of some feature dimensions (Xu et al., 2019). It is still unclear how to coordinate the two filtering methods so as to achieve superior performance for a wide range of graphs.

To address this issue, we propose a novel filter-synergy approach to enhance the existing low-pass filtering based continuous models. Building upon previous research (Xhonneux et al., 2020), our method leverages the link between diffusion dynamics and continuous messaging in GNNs to theoretically derive the high-pass filtering based continuous graph neural network, which can enjoy the deep architecture and be nearly free of over-smoothing. To coordinate the low and high pass filtering results (i.e., learned representations at steady states), we introduce a dimension masking method, which serves as a feature selector to learn the task-specific feature dimensions, enabling the model to be better adaptive to a full spectrum of heterophily of the graph structure. We show the state-of-the-art performance of our model over both strong discrete and continuous models on the benchmark datasets.

In summary, the main contributions of our work are as follows:

- We introduce a high pass filtering based continuous graph neural network model, which can bring significant gains to supervised node classification on graphs with strong heterophily.
- To exploit the advantages of high-pass filtering, we propose a novel feature aggregation method to fuse features from low-pass and high-pass networks respectively by leveraging masking mechanism at dimension level, which can flexibly coordinate the two types of features and offer more distinguishable representations for each class of nodes.

2. Preliminaries

Notation. Suppose we have an undirected connected graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ with f-dimensional attributes $\boldsymbol{X} \in \mathbb{R}^{n\times f}$ on the nodes, where V is the set of nodes, $\boldsymbol{E}$ is the set of edges, and $n = |\boldsymbol{V}|$ is the number of nodes. The adjacency matrix associated with graph G is denoted as $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. Let $\boldsymbol{D} = diag(d_i)$ be the diagonal degree matrix. The Graph Laplacian is defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, which is Symmetric Positive Semi-Definite (SPSD) (Chung, 1997). Through the eigenvalues and eigenvectors, we have $\boldsymbol{L} = UVU^T$, where $V = diag([\lambda_1, \lambda_2, \cdots, \lambda_n])$. In addition to L, some other variants are also commonly used, such as the symmetric normalized Laplacian $\boldsymbol{L}_{sym} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}$ and the random walk normalized Laplacian $\boldsymbol{L}_{rw} = \boldsymbol{D}^{-1}\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A}$. Just as graph Laplacians and their variants are viewed as high-pass filters for graph signals, similarly, low-pass filters for graph signals can be seen as $\boldsymbol{A}_{rw} = \boldsymbol{I} - \boldsymbol{L}_{rw} = \boldsymbol{D}^{-1}\boldsymbol{A}, \boldsymbol{A}_{sym} = \boldsymbol{I} - \boldsymbol{L}_{sym} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}$. Because they are real symmetric matrix, their eigenvalues satisfy $\lambda_i(\boldsymbol{A}_{rw}) = \lambda_i(\boldsymbol{A}_{sym}) \in (-1, 1]$.Then, the graph with self-loop at every node can be represented as $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$, and the corresponding diagonal degree matrix is $\widetilde{\boldsymbol{D}} = \boldsymbol{D} + \boldsymbol{I}$. Thus, the self-looped adjacency can be symmetrically normalized as $\boldsymbol{A_{sym}} = \widetilde{\boldsymbol{D}}^{\boldsymbol{-1/2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{\boldsymbol{-1/2}}$ and $\boldsymbol{L_{sym}} = \boldsymbol{I} - \widetilde{\boldsymbol{D}}^{\boldsymbol{-1/2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{\boldsymbol{-1/2}}$.
GNN. The renormalized affinity matrix basically adds a self-loop to each node in the graph, and is widely used in GCN (Kipf and Welling, 2017) as follows:

$$\boldsymbol{H} = \widehat{A}_{sym} ReLU(\widehat{\boldsymbol{A}}_{\boldsymbol{sym}}\boldsymbol{X}\boldsymbol{W}^{(0)})\boldsymbol{W}^{(1)}, \tag{1}$$

where $\boldsymbol{W}^{(0)}$ is a layer-specific trainable weight matrix, which can be learned by minimizing the cross-entropy between ground truth and predicted labels on the training set $\boldsymbol{T}$, as

$$\boldsymbol{L} = -\sum_{v_i \in T}\sum_{k=1}^{C} y_{ik} \ln z_{ik}, \tag{2}$$

in which $z = softmax(\boldsymbol{H})$ is the output of last layer.
Neural ODEs. In this paper, we adopt a continuous Ordinary Differential Equation (ODE) formulation, which is expressed as follows:

$$x(T) = x(0) + \int_0^T \frac{\partial x(t)}{\partial t} dt, \qquad \frac{\partial x(t)}{\partial t} = F(x(t), t). \tag{3}$$

Here, x can be a scalar, vector-valued, or matrix-valued variable, and F is a function that we parameterize to define the hidden dynamics. The work by (Chen et al., 2018) demonstrated how it is possible to perform backpropagation through such an ODE equation, enabling us to use it as a fundamental building block for a neural network.

3. Model

In this section, we present our method, which aims to improve the learning of node representations for supervised node classification tasks using high-pass and low-pass filters. Our approach involves several key steps. First, we apply a linear transformation layer, denoted as $f_\theta(\cdot)$, to map the features of all nodes into a latent space, represented as $\boldsymbol{H}(0) = f_\theta(\boldsymbol{X})$. This transformation helps capture the initial representations of nodes. Next, we employ

Ordinary Differential Equations (ODEs) to define the continuous dynamics of node representations. ODEs enable the efficient capture of long-term dependencies between nodes. To fully leverage these dependencies, we design a low-pass filter, denoted as $\boldsymbol{A}_{sym}$, and a high-pass filter, denoted as $\boldsymbol{L}_{sym}$, to separate the low-pass and high-pass signals from the node representations. The separation of low-pass and high-pass signals allows us to design targeted ODEs that not only define the continuous dynamics of node representations but also possess discriminative properties. We construct two ODEs based on different frequencies, drawing inspiration from existing diffusion-based graph methods. In both ODEs, each feature channel (i.e., dimension) represented by a node at a given state evolves independently, taking into account the fusion of label information with frequency information. This design choice enables better integration and utilization of the node representations for downstream tasks. A crucial step in our framework is the establishment of a coordination masker. This masker takes advantage of the complementarity between high-pass and low-pass filters. By applying the mask to the node representations ($\boldsymbol{H}_L(t)$ and $\boldsymbol{H}_H(t)$) obtained at the end time t, the representations can be fully integrated and effectively utilized for downstream tasks. Overall, our method combines ODEs, high-pass and low-pass filters (see Section 3.1), and filtering coordination to improve node representation learning for supervised node classification tasks (see Section 3.2).

3.1. Low and High-Pass Filtering

Since the nodes in the graph are connected to each other and the attributes or categories of the nodes may be different, more detailed operations are performed when disseminating information between different nodes. Some works (Bo et al., 2021; Huang et al., 2023b) proposes low-pass and high-pass filter effects to make nodes more similar and dissimilar, respectively. Inspired by the existing graph diffusion-based method CGNN (Xhonneux et al., 2020), for the representation propagation process, we consider information diffusion from two channels, i.e., low-pass and high-pass:

$$\begin{aligned}\boldsymbol{H}_L^{(n+1)} &= \sum_{n}^{i=0} \widehat{\boldsymbol{A}}_{sym}\boldsymbol{H}^{(0)} = (\widehat{\boldsymbol{A}}_{sym} - \boldsymbol{I})^{-1}(\widehat{\boldsymbol{A}}_{sym}^{n+1} - \boldsymbol{I}),\\ \boldsymbol{H}_H^{(n+1)} &= \sum_{n}^{i=0} \widehat{\boldsymbol{L}}_{sym}\boldsymbol{H}^{(0)} = (-\widehat{\boldsymbol{A}}_{sym})^{-1}((\boldsymbol{I} - \widehat{\boldsymbol{A}}_{sym})^{n+1} + \boldsymbol{I}).\end{aligned} \tag{4}$$

where we see that the representation $\boldsymbol{H}_L^{(n+1)}, \boldsymbol{H}_H^{(n+1)} \in \mathbb{R}^{|V|\times d}$ at step $n+1$ incorporates all the information propagated up to $n+1$ steps with the initial representation $\boldsymbol{H}^{(0)} = f_\theta(\boldsymbol{X})$. Intuitively, each node at stage $n+1$ learns the node information from its neighbours through $\boldsymbol{H}_L^{(n+1)} = \widehat{\boldsymbol{A}}_{sym}\boldsymbol{H}_L^{(n)} + \boldsymbol{H}^{(0)}, \boldsymbol{H}_H^{(n+1)} = \widehat{\boldsymbol{L}}_{sym}\boldsymbol{H}_H^{(n)} + \boldsymbol{H}^{(0)}$ and remembers its original node features through $\boldsymbol{H}^{(0)}$. This allows us to learn the graph structure without forgetting the original node features.

In view of the previous research (Klicpera et al., 2019; Xhonneux et al., 2020), we can clearly understand the effectiveness of the discrete propagation process in Eq. 4, so we replace n with a continuous variable $t \in \mathbb{R}_0^+$, convert the discrete form into a continuous diffusion form and use ODE to characterize this continuous spread the dynamics. We treat the summation in Eq. 4 as the Riemann sum of the integral from time 0 to time $t = n+1$, which allows us to move naturally from discrete propagation processes to the continuous case, where we separate the low-pass filtering and high-pass filtering has undergone a transformation, as follow:

$$\begin{aligned}\frac{\partial \boldsymbol{H}_L(t)}{\partial t} &= ln\widehat{\boldsymbol{A}}_{\boldsymbol{sym}}\boldsymbol{H}_L(t) + \boldsymbol{H}(0),\\ \frac{\partial \boldsymbol{H}_H(t)}{\partial t} &= ln\widehat{\boldsymbol{L}}_{\boldsymbol{sym}}\boldsymbol{H}_H(t) + \boldsymbol{H}(0) = ln(\boldsymbol{I} - \widehat{\boldsymbol{A}}_{\boldsymbol{sym}})\boldsymbol{H}_H(t) + \boldsymbol{H}(0),\end{aligned} \tag{5}$$

where $\boldsymbol{H}(0) = f_\theta(X)$ is the output of the encoder $f_\theta(\cdot)$. We provide proof in the appendix A. In practice, $ln(\widehat{\boldsymbol{A}}_{sym}), ln(\widehat{\boldsymbol{L}}_{sym})$ in Eq. 5 is intractable to compute, hence we approximate it using the first order of the Taylor expansion, i.e. $ln(\widehat{\boldsymbol{A}}_{sym}) \approx \widehat{\boldsymbol{A}}_{sym} - \boldsymbol{I}$, $ln(\boldsymbol{I} - \widehat{\boldsymbol{A}}_{sym}) \approx -\widehat{\boldsymbol{A}}_{sym}$ which gives us:

$$\frac{\partial \boldsymbol{H}_L(t)}{\partial t} = (\widehat{\boldsymbol{A}}_{\boldsymbol{sym}} - \boldsymbol{I})\boldsymbol{H}_L(t) + \boldsymbol{H}(0), \quad \frac{\partial \boldsymbol{H}_H(t)}{\partial t} = (-\widehat{\boldsymbol{A}}_{\boldsymbol{sym}})\boldsymbol{H}_H(t) + \boldsymbol{H}(0). \tag{6}$$

These are the two ODEs used in our model EGLD. We can find that the process of continuous diffusion at high-pass can actually be regarded as the process of negative sign diffusion at low-pass. The two ODEs we use can be understood theoretically. Specifically, the node representation matrix $\boldsymbol{H}(t)$ at time t has an analytical form, and its form expresses the following proposition.

Proposition 1 *The analytical solution of the ODEs defined in Eq. 6 are given by:*

$$\begin{aligned}\boldsymbol{H}_L(t) &= (\widehat{\boldsymbol{A}}_{\boldsymbol{sym}} - \boldsymbol{I})^{-1}(e^{(\widehat{\boldsymbol{A}}_{sym} - \boldsymbol{I})t} - \boldsymbol{I})\boldsymbol{H}(0) + e^{(\widehat{\boldsymbol{A}}_{sym} - \boldsymbol{I})t}\boldsymbol{H}(0),\\ \boldsymbol{H}_H(t) &= (\widehat{\boldsymbol{L}}_{\boldsymbol{sym}} - \boldsymbol{I})^{-1}(e^{(\widehat{L}_{sym} - \boldsymbol{I})t} - \boldsymbol{I})\boldsymbol{H}(0) + e^{(\widehat{L}_{sym} - \boldsymbol{I})t}\boldsymbol{H}(0).\end{aligned} \tag{7}$$

We provide proof in the appendix. From the proposition, since the eigenvalues of $(\widehat{\boldsymbol{A}}_{\boldsymbol{sym}} - \boldsymbol{I})$ and $-\widehat{\boldsymbol{A}}_{\boldsymbol{sym}}$ are in the interval $[-2, 0)$ and $(-1, 1]$, we can obtain the values of $\boldsymbol{H}_L(t)$ and $\boldsymbol{H}_H(t)$ changing with time t respectively.

Proposition 2 *If t to ∞, $\boldsymbol{H}_L(t)$ can be viewed as the sum of all different orders of propagation information, essentially having an infinite number of discrete propagation layers, $\boldsymbol{H}_H(t)$ can be seen as suppressing low-pass signals when the eigenvalue λ is at $(0, 1)$, and when λ is at $(1, 2]$, it makes the high eigenvalues achieve negative feedback.*

Let t to ∞. Then we can approximate $\boldsymbol{H}_L(t)$, $\boldsymbol{H}_H(t)$ as:

$$\begin{aligned}\boldsymbol{H}_L(t) &\approx (\boldsymbol{I} - \widehat{\boldsymbol{A}}_{\boldsymbol{sym}})^{-1}\boldsymbol{H}(0) \approx (\sum_{i=0}^{\infty} \widehat{\boldsymbol{A}}^{\boldsymbol{i}}_{\boldsymbol{sym}})\boldsymbol{H}(0),\\ \boldsymbol{H}_H(t) &\approx (\boldsymbol{I} - \widehat{\boldsymbol{L}}_{\boldsymbol{sym}})^{-1}\boldsymbol{H}(0) \approx (\widehat{\boldsymbol{A}}_{\boldsymbol{sym}})^{-1}\boldsymbol{H}(0).\end{aligned} \tag{8}$$

These are approximated by t time to ∞, $lim_{t\to\infty} e^{(\widehat{\boldsymbol{A}}_{\boldsymbol{sym}} - I)t} = 0$, $lim_{t\to\infty} e^{(-\widehat{\boldsymbol{A}}_{\boldsymbol{sym}} - I)t} = 0$.

3.2. Filtering Coordination

In fact, the representations propagated between nodes in different frequency states are very different, how can they be well combined and used in downstream tasks? Generally, addition (i.e., summing) and concatenation fusion are commonly used. Previous study (Xu et al., 2019) has shown that simple addition or global mixing is very easy to cover up important information and cause a lot of noise information. Although the concatenation method provides more information than addition, it is easy to cause redundancy of information and affect the representation.

To this end, we design a masking method for dimension reduction, based on the *coordination principle*: encouraging the consistency of feature dimensions on the nodes in the same class. Moreover, due to the complementarity of high and low pass filtering, representations in different states are complementary masked. First, we obtain binarized mask information through the representations $H_L(t)$ and $H_H(t)$ at steady states of the CGNNs from two frequency channels. In order to enable the model to perform gradient propagation, we use activation function operations to perform approximations to achieve the effect of the mask, which reads as

$$\boldsymbol{M} = Sigscal([\boldsymbol{H}_L(t_1), \boldsymbol{H}_H(t_1)]\boldsymbol{W}_1 + b_1), \tag{9}$$

where $\boldsymbol{M} \in \mathbb{R}^{n\times d}$, $\boldsymbol{W}_1 \in \mathbb{R}^{2d\times d}$, $Sigscal(\cdot) = Sigmoid(\cdot\frac{1}{\tau})$ is to use τ to scale the value to approximate the mask effect (i.e. 0, 1). The $[\boldsymbol{H}_L(t_1), \boldsymbol{H}_L(t_1)]$ indicates the concatenation of the two representations $\boldsymbol{H}_L(t_1)$ and $\boldsymbol{H}_H(t_1)$. Here we simply use a linear projection with bias b_1. After obtaining the masks for feature dimension reduction, we can create the final node representations by fusing the masked features

$$\boldsymbol{H}_{final} = f_\theta([\boldsymbol{M} \odot \boldsymbol{H}_L(t_1), (1 - \boldsymbol{M}) \odot \boldsymbol{H}_H(t_1)]). \tag{10}$$

Here, we demonstrate how to implement the coordination principle. We perform masking regularization w.r.t classification by introducing label information. It is expected that the masking on the nodes with the same label should be in the same dimension, and vice versa, as follows:

$$\boldsymbol{\mathcal{L}}_{dis} = ||\boldsymbol{G}\boldsymbol{G}^T - I||_F^2, \quad \boldsymbol{G}^T = softmax((\boldsymbol{M}_{|S|}^T\boldsymbol{B})), \tag{11}$$

where the label information $\boldsymbol{B} = one - hot(\boldsymbol{Y}_{|S|})\boldsymbol{B} \in \mathbb{R}^{|S|\times c}, \boldsymbol{Y}_{|S|}$ represents the training set label, $|S|$ is the number of training set labels, and then combine the mask matrix $\boldsymbol{M}_{|S|}$ to obtain the mask dimension information of each category, $\boldsymbol{G} \in \mathbb{R}^{c\times d}$ represents the distribution of mask position dimensions for each category, $||\cdot||_F$ indicates the Frobenius norm. In this way, the final representations will be more conducive to the classification of nodes.

3.3. Objective

Armed with the key model components introduced in the previous two sections, i.e. high and low pass filtering and dimension coordination, we next introduce our jointly optimized classification loss and masking regularization for feature coordination. The overall objective function can be stated as:

$$\boldsymbol{L} = \boldsymbol{L}_{class} + \lambda \boldsymbol{L}_{dis} = -\sum_{v_i \in T} \sum_{k=1}^{C} y_{ik} \ln z_{ik} + \lambda ||\boldsymbol{G}\boldsymbol{G}^T - I||_F^2, \tag{12}$$

where $z = Softmax(H_{final})$, λ is the hyper-parameters that control the contribution of $\mathcal{L}_{dis}$.

The computational complexity of the Eq.6 is $O(2|E|dn_t)$, where n_t, $|E|$ and d are number of time steps in time interval $[0, T]$, number of edges and number of feature dimension, respectively. The complexity of the mask part of the fused representation is $O(2nd^2 + 2ndc + |U|dc + c^2d^2)$, where $|U|$ and c are the number of nodes in the training set and the number of label categories, respectively. So the complexity of EGLD is $O(|E|dn_t + nd^2)$, which is linearly related to the number of nodes and edges.

Table 1: Mean accuracy±stdev over different data splits on the ten datasets. The best performance for each dataset is highlighted in bold and the second best performance is underlined for comparison.

Dataset	**Chameleon**	**Squirrel**	**Film**	**Texas**	**Wisconsin**	**Cornell**	**Cora**	**Citeseer**	**Pubmed**	
Hom level	0.23	0.22	0.22	0.11	0.21	0.30	0.81	0.74	0.8	**Avg.**
#Nodes	2,277	5,201	7,600	183	251	183	2,708	3,327	18,717	**Rank**
#Edges	31,421	198,493	26,752	295	466	280	5,278	4,676	44,327	
#Classes	5	5	5	5	5	5	6	7	3	
MLP	46.93±1.7	29.95±1.6	34.78±1.2	79.19±6.3	83.15±5.7	79.79±4.2	75.13±2.7	73.26±1.7	85.69±0.3	18.6
GCN	65.92±2.5	49.78±2.0	27.51±1.2	55.14±5.16	51.76±3.06	60.54±5.3	86.98±1.27	76.50±1.36	88.42±0.5	17.7
GAT	65.32±1.9	46.79±2.0	29.03±0.9	52.16±6.63	49.41±4.09	61.89±5.05	87.30±1.10	76.55±1.23	86.33±0.48	15.6
GraphSAGE	58.71±2.3	41.05±1.1	34.37±1.3	82.70±5.9	81.76±5.6	75.59±5.2	86.60±1.8	75.61±1.6	88.01±0.8	16.9
MixHop	60.50±2.53	43.80±1.48	32.22±2.34	77.84±7.73	75.88±4.90	73.51±6.34	87.61±0.85	76.26±1.33	85.31±0.61	16.8
GCNII	63.86±3.04	36.37±1.6	34.40±0.7	77.57±3.8	80.39±3.4	77.86±3.7	**88.37±1.25**	77.33±1.48	**90.15±0.43**	11.4
H2GCN	59.56±1.8	37.90±2.0	35.55±1.6	82.16±5.2	85.88±4.3	82.16±6.0	88.13±1.4	76.73±1.4	88.46±0.7	12.9
Geom-GCN	60.90±2.8	38.14±0.92	31.63±1.15	60.18	67.57	64.12	85.35±1.57	**78.02±1.15**	89.95±0.47	15.9
FAGCN	45.13±2.2	31.77±2.1	34.51±0.7	72.43±5.6	67.84±4.8	77.06±6.3	87.87±0.8	76.76±1.6	88.80±0.6	16.9
GPRGNN	46.58±1.71	31.61±1.24	34.63±1.22	78.38±4.36	82.94±4.21	80.27±8.11	87.95±1.18	77.13±1.67	87.54±0.38	15.4
GGCN	71.14±1.84	55.17±1.58	37.54±1.56	84.86±4.55	86.86±3.29	85.68±6.63	87.95±1.05	77.14±1.45	89.15±0.37	6.2
Diag-NSD	68.68±1.73	54.78±1.81	37.79±1.01	85.67±6.95	88.63±2.75	86.49±7.35	87.14±1.06	77.14±1.85	89.42±0.13	5.8
O(d)-NSD	68.04±1.58	56.34±1.32	**37.81±1.15**	85.95±5.51	89.41±4.74	84.86±4.71	86.90±1.13	76.77±1.57	89.49±0.40	6.3
Gen-NSD	67.93±1.58	53.17±1.31	37.80±1.22	82.97±5.13	89.21±3.84	85.68±6.51	87.30±1.15	76.32±1.65	89.33±0.35	8.2
ACM-GCN	66.93±1.85	54.40±1.88	36.28±1.09	**87.84±4.40**	88.43±3.22	85.14±6.07	87.91±0.95	77.32±1.70	90.00±0.52	5.8
AERO-GNN	71.58±2.4	61.76±2.4	36.57±1.1	84.35±5.2	84.80±3.3	81.24±6.8	-	-	-	-
BLEND	60.11±2.09	43.06±1.39	35.63±0.89	83.24±4.65	84.12±3.56	85.95±6.82	88.09±1.22	76.63±1.60	89.24±0.42	10.3
GRAND	54.63±2.54	40.05±1.50	35.62±1.01	75.68±7.25	79.41±3.64	82.16±7.09	87.36±0.96	76.46±1.77	89.02±0.51	15.2
CGNN	46.89±1.66	29.24±1.09	35.95±0.86	81.35±4.05	74.31±7.26	66.22±7.69	87.10±1.35	76.91±1.81	87.70±0.49	17.0
Cont Diag-NSD	62.06±3.84	38.17±9.29	36.85±1.21	82.97±4.73	86.47±2.55	80.00±6.07	86.88±1.21	76.56±1.19	89.47±0.42	12.0
Cont O(d)-NSD	63.18±1.69	40.40±2.01	36.39±1.37	82.43±5.95	84.50±4.34	72.16±10.40	86.70±1.24	75.19±1.67	89.12±0.30	14.7
Cont Gen-NSD	66.40±2.28	52.57±2.76	37.28±0.74	83.78±6.62	85.29±3.31	84.60±4.69	87.45±1.35	77.54±1.72	89.67±0.40	7.2
CDE-GraphBel	56.34±3.2	39.19±2.12	35.88±0.9	87.57±3.24	87.84±4.87	85.14±5.95	84.54±1.48	76.36±1.8	88.61±0.42	13.3
GREAD-FB	65.83±1.10	50.57±1.52	37.70±0.51	87.03±3.97	88.04±1.63	85.95±5.64	88.01±0.80	77.42±1.93	90.08±0.46	5.6
EGLD	**76.86±1.90**	**68.83±2.14**	36.43±0.74	87.07±2.92	**89.54±2.91**	**86.55±5.80**	88.20±1.47	77.41±1.59	89.74±0.50	**2.6**

4. Experiment

In order to verify the effectiveness of the continuous model we designed using two Laplacian flows for information propagation fusion, we conduct experiments to answer the following questions:

Q1 Is our model superior to the existing strong baselines, including continuous or discrete

methods?
Q2 Can we solve the problems of deepening convolution layers without over-smoothing and achieving good performance on graphs with strong heterophily at the same time?
Q3 Is it meaningful to introduce high-pass filtering, and what role does it play compared to low-pass filtering?

4.1. Node classification

To answer **Q1**, we evaluate the node classification performance of the proposed mothed and compare it with state-of-the-art heterophily-oriented GNN models on heterophilious graphs. Moreover, we test our model on some benchmark graph datasets with strong homophily that cover a full spectrum of heterophily.

Datasets. We evaluate the performance of EGLD model and existing GNNs in node classification on various real-world datasets (Sen et al., 2008; Pei et al., 2020; Rozemberczki et al., 2021; Tang et al., 2009; Yang et al., 2016). For all benchmarks, we use the feature vectors, class labels, and 10 random splits (48%/32%/20% of nodes per class for train/validation/test[1]) from (Pei et al., 2020).

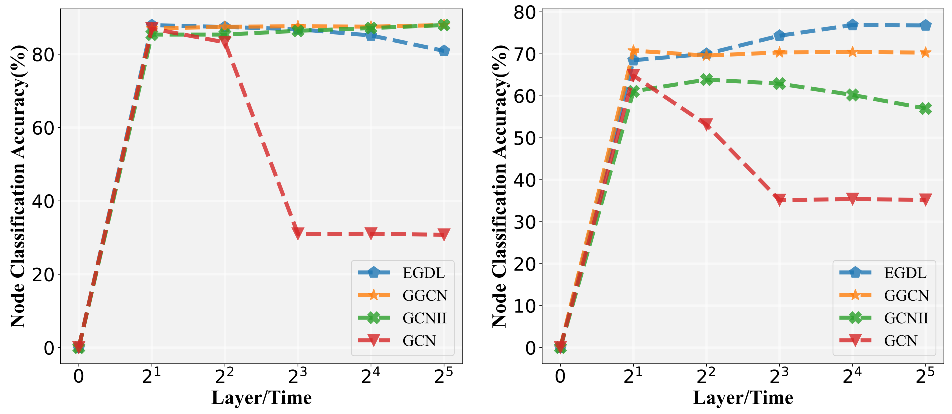
Baselines. We compare our scheme with following methods, some of which are shown to be competitive on various graphs: Multilayer Perceptron (MLP), SGC (Wu et al., 2019), Graph Convolutional Network (GCN) (Kipf and Welling, 2017), Graph Attention Network(GAT) (Velickovic et al., 2018), Mixhop (Abu-El-Haija et al., 2019) and GCNII (Chen et al., 2020). For the heterophilious datasets, we specifically compare our model with heterophily-oriented methods, namely, two variants of H2GCN (i.e., H2GCN-1 and H2GCN-2) (Zhu et al., 2020), Geom-GCN (Pei et al., 2020), FAGNN (Bo et al., 2021) and one variant of GCNII (Chen et al., 2020) wherein parameters are shared between layers, GPRGNN (Chien et al., 2021), GGCN (Yan et al., 2022), ACM-GCN (Luan et al., 2022), AERO-GNN (Lee et al., 2023). Additionally we compare to recent ODE-based GNN models, Continuous Graph Neural Networks (Xhonneux et al., 2020), GRAND (Chamberlain et al., 2021a), BLEND (Chamberlain et al., 2021b), Neural Sheaf Diffusion (Bodnar et al., 2022)which includes continuous and discrete three variant models, CDE-GraphBel (Zhao et al., 2023), GREAD-FB (Choi et al., 2023).

Model Setting. We implement the proposed EGLD and some necessary baselines using PyTorch and PyTorch Geometric, a library for deep learning on irregularly structured data built upon PyTorch. We try our best to provide a rigorous and fair comparison between different models. To mitigate the effects of randomness, we run each method 10 times and report the average performance. For the baseline methods, whose results on the benchmark datasets are publicly available, we directly present the results. For the models without publicly reported results, we use the original codes published by their authors and fine-tune them.

Experimental Results. To answer **Q1**, we present the test accuracy results of different GNN models on the supervised node classification task across datasets with varying homophily levels. The results are summarized in Table 1. From the table, we observe that our proposed EGLD model achieves new state-of-the-art performance on almost all

1. (Pei et al., 2019) claims that the ratios are 60%/20%/20%, which are different from the real data splits shared on GitHub.

heterophilous graphs (where $h < 0.5$), surpassing the best-performing existing models. Remarkably, EGLD also demonstrates strong performance on homophily graphs. It outperforms most methods across all datasets, as evidenced by its average rank of 2.3. This indicates the model's exceptional adaptability to graphs at various homophily levels. Furthermore, EGLD achieves competitive accuracy when compared to leading GNNs designed specifically for homophilous graphs. It is worth noting that, in previous research (Bodnar et al., 2022) and based on experimental results, continuous models typically exhibit inferior performance compared to discrete models. However, our continuous model, EGLD, demonstrates a significant improvement in performance compared to the ODE-based GNN model. This improvement can be attributed to the introduction of high-pass filtering and the effective coordination of the final representations.



(a) Cora (b) Chameleon

Figure 1: Model performance at different layers or end times.

Table 2: Ablation study.

Datasets	Cora	CiteSeer	Pubmed	Chameleon	Squirrel	Film	Connell	Wisconsin	Texas
w/o L-F	30.94	23.91	68.41	61.16	51.36	31.17	62.70	72.35	74.05
w/o H-F	87.66	76.84	87.17	54.05	41.37	30.82	61.62	57.05	62.97
w/o C-Mask	87.80	77.00	89.23	73.68	56.73	36.01	85.40	86.86	85.79
EGLD	**88.20**	**77.41**	**89.74**	**76.86**	**68.83**	**36.43**	**86.55**	**89.54**	**87.07**

4.2. Depth and PDE Solvers

There are various numerical methods (Euler, RK4, DOPRI5) for solving the diffusion equation, and these solvers can be divided into single-step and multi-step schemes, where multi-step schemes involve using multiple function evaluations at different time steps to compute the next iteration. We compared running EGLD using the Euler method, DOPRI5 and the adaptive Runge-Kutta 4 method. Finally, our model chooses the Euler solver with better performance and computational efficiency. To answers **Q2**, our model can avoid

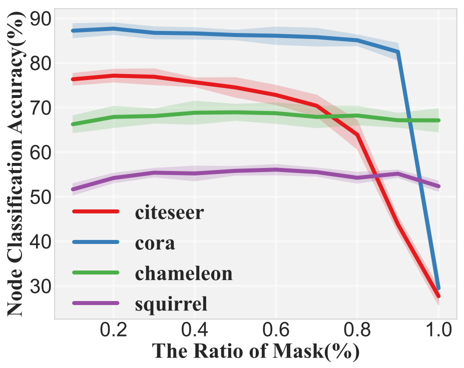


Figure 2: The performance of the model with random mask high and low frequency dimension information.

the oversmoothing problem, we experimentally show the performance of different methods with different numbers of layers (e.g., GGCN and GCNII) or end times T (e.g., EDLD), we performed an experiment using the Euler, varying the integration time T while holding the other hyper-parameters fixed. As shown in the Figure 1, GCN (red) achieves the best performance at the second layer under both Cora and Chameleon. As the number of layers increases, over-smoothing occurs and the performance drops sharply. However, the performance of the two models GGCN (yellow) and GCNII (green) that perform well in alleviating oversmoothing remains stable as the number of layers increases. Compared with our model EGLD (blue), as the time step increases, the performance under Cora first increases and then has a small decline, while the performance under Chameleon maintains an upward trend. This is just like the theorem we analyzed, when $t \to \infty$ in the high-pass filtering, the low-frequency signal has a tendency to amplify to infinity, and the high eigenvalues part presents a negative feedback effect. As a result, our model has a small performance drop on homophily graphs when the time step is relatively large, but it can tend to be stable or even slightly improved on heterophilious graphs.

4.3. Ablation Study

To answer **Q3**, we performed a series of experiments and analyses. Firstly, we removed the filtering coordination from the EGLD model and instead applied the mask method on random dimensions to complementarily fuse low-pass and high-pass information. Figure 2 illustrates the results, where the x-axis represents the ratio of dimensions covered by low-pass information (m, the number of covered dimensions divided by the total number of dimensions), and the ratio of dimensions covered by high-pass information $(1 - m)$. The figure shows that as the proportion of low-pass masks increases, the performance of the model on homogeneous graph data (i.e. cora and citeseer) gradually declines. In contrast, the performance on heterophily graph data (i.e. chameleon and squirrel) initially improves and then decreases. Notably, at a small proportion ($m = 0.2$), the performance on homogeneous graphs exhibits a slight improvement, demonstrating the effectiveness of high-pass information in conjunction with low-pass information.

Next, we conducted ablation experiments to evaluate the effectiveness of different components in our model: low-pass filtering (L-F) and high-pass filtering (H-F), corresponding to the first and second terms in the Eq.6, and label-based mask constraint (C-L) in the Eq.11. These experiments were conducted separately on homophily and heterophily datasets. The Table 2 clearly indicate that removing high-pass filtering leads to a significant drop in performance on heterophily graphs, emphasizing the importance of high-pass filter in handling high heterogeneity. Similarly, removing low-pass filtering significantly reduces the performance on all datasets, suggesting that low-pass filter play a dominant role in representation learning. Additionally, we analyzed the impact of removing the filtering coordination. By removing this constraint, the mask becomes more random, resulting in insufficient information for fusion and utilization. Consequently, the model's performance slightly declines. These findings demonstrate that the effective utilization of high-pass information are crucial for model performance.

In summary, our experimental results show that low-pass filtering dominates in graph representation learning, while high-pass filtering is effective not only in heterophily data, but also in homophily data. The key is to effectively coordinate low-pass and high-pass information.

5. Related Work

Oversmoothing and Heterophily. In some early works, different methods were used to obtain and aggregate high-order neighbor information to adapt normal GNNs to heterophily networks. For example, H2GCN (Zhu et al., 2020) spliced high-order neighbors, Geom-GCN (Pei et al., 2020) aggregated distant nodes in latent space, and CPGNN (Zhu et al., 2021) used compatibility matrix to learn the likelihood of connections between nodes in different categories. Some recent works have jointly solved the two problems of oversmoothing and heterophily. For example, GPR-GNN (Chien et al., 2021) learns a weight for each step of propagation. GGCN (Yan et al., 2022) analyzes the performance of linear SGC (Wu et al., 2019) on random attribute maps. Sheaf (Bodnar et al., 2022) uses cell bundle theory and focuses on diffusion. Common to their work on partial differential equations is the discovery of negative-signed side benefits in the propagation process. However, in our work, using high and low frequency filtering to propagate information can also play the same role. Although FAGCN (Bo et al., 2021) and ACM-GCN (Luan et al., 2022) both integrate low-pass and high-pass filter. In contrast, our work mainly focuses on using graph partial differential equations to simulate message propagation and close labels at the time of fusion.

Neural ODEs. Since (Chen et al., 2018) introduction of neural ODEs, this topic has become an emerging field, and subsequent works (Dupont et al., 2019; Finlay et al., 2020) have explored and extended it into new areas. In many follow-up works, neural ODE was applied to GNN: (Avelar et al., 2021) proposed to use continuous residual module for GNN; (Xhonneux et al., 2020) extending GNN to continuous time is the solution of constant linear diffusion partial differential equation; GRAND (Chamberlain et al., 2021a) regards GNN as a continuous diffusion process, and spread through the graph diffusion equation; sheaf introduces the cell beam diffusion operator to control the diffusion process; ACMP (Wang et al., 2022) is inspired by the particle reaction-diffusion process, and models the repulsive and gravitational interactions as dual flow directions between nodes. (Zhao et al., 2023)incorporates the principle of heterogeneity by using the convection-diffusion equation (CDE) to model information flow on nodes.

6. Conclusion

This paper presents a novel exploration of continuous high-pass filtering and establishes connections between high-pass and low-pass filtering. Moreover, we investigate the coordination method between these continuous processes at feature dimension level, aiming to improve adaptation to downstream tasks on a variety of graph structures. Through empirical experiments, we demonstrate the superior efficacy of our approach compared to the existing strong methods. Additionally, we validate the effectiveness of each component in our method. We note that our approach essentially involves constructing a dual-channel continuous method. However, as part of future work, we intend to enhance model efficacy by obtaining high and low frequency information via a single continuous process. This improvement will streamline the computational complexity of our model while maintaining its effectiveness.

7. Acknowledgements

This work is supported by the National Natural Science Foundation (NSFC 62276099) and SWPU Innovation Base funding (No.642).

References

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning*, pages 21–29, 2019.

Pedro H. C. Avelar, Anderson R. Tavares, Marco Gori, and Luís C. Lamb. Discrete and continuous deep residual learning over graphs. In *ICAART (2)*, pages 119–131. SCITEPRESS, 2021.

Anson Bastos, Abhishek Nadgeri, Kuldeep Singh, Isaiah Onando Mulang, Saeedeh Shekarpour, Johannes Hoffart, and Manohar Kaul. Recon: relation extraction using knowledge graph context in a graph neural network. In *Proceedings of the Web Conference*, pages 1673–1685, 2021.

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3950–3957, 2021.

Cristian Bodnar, Francesco Di Giovanni, Benjamin Chamberlain, Pietro Liò, and Michael Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *Advances in Neural Information Processing Systems*, 35:18527–18541, 2022.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, pages 1407–1418. PMLR, 2021a.

Benjamin Chamberlain, James Rowbottom, Davide Eynard, Francesco Di Giovanni, Xiaowen Dong, and Michael Bronstein. Beltrami flow and neural diffusion on graphs. *Advances in Neural Information Processing Systems*, 34:1594–1609, 2021b.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pages 1725–1735, 2020.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31, 2018.

E. Chien, J. Peng, P. Li, and O. Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

Jeongwhan Choi, Seoyoung Hong, Noseong Park, and Sung-Bae Cho. GREAD: graph neural reaction-diffusion networks. In *ICML*, volume 202, pages 5722–5747, 2023.

Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *Advances in Neural Information Processing Systems*, 32, 2019.

Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. How to train your neural ode: the world of jacobian and kinetic regularization. In *International Conference on Machine Learning*, pages 3154–3164. PMLR, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272, 2017.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

Jincheng Huang, Lun Du, Xu Chen, Qiang Fu, Shi Han, and Dongmei Zhang. Robust mid-pass filtering graph convolutional networks. In *Proceedings of the ACM Web Conference 2023*, page 328–338, 2023a.

Jincheng Huang, Ping Li, Rui Huang, Na Chen, and Acong Zhang. Revisiting the role of heterophily in graph representation learning: An edge classification perspective. *ACM Transactions on Knowledge Discovery from Data*, 2023b.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International Conference on Machine Learning*, pages 4839–4848, 2020.

Qiyu Kang, Yang Song, Qinxu Ding, and Wee Peng Tay. Stable neural ode with lyapunov-stable equilibrium points for defending against adversarial attacks. *Advances in Neural Information Processing Systems*, 34:14925–14937, 2021.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations*, 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *7th International Conference on Learning Representations*, 2019.

Soo Yong Lee, Fanchen Bu, Jaemin Yoo, and Kijung Shin. Towards deep attention in graph neural networks: Problems and remedies. In *ICML*, volume 202, pages 18774–18795, 2023.

Yaguang Li, Chuizheng Meng, Cyrus Shahabi, and Yan Liu. Structure-informed graph auto-encoder for relational inference and simulation. In *ICML Workshop on Learning and Reasoning with Graph-Structured Data*, volume 8, page 2, 2019.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. In *NeurIPS*, 2022.

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

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *8th International Conference on Learning Representations*, 2020.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Yang Song, Qiyu Kang, and Wee Peng Tay. Error-correcting output codes with ensemble diversity for robust learning in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9722–9729, 2021.

Yang Song, Qiyu Kang, Sijie Wang, Kai Zhao, and Wee Peng Tay. On the robustness of graph neural diffusion to topology perturbations. *Advances in Neural Information Processing Systems*, 35:6384–6396, 2022.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 807–816, 2009.

Sijing Tu and Stefan Neumann. A viral marketing-based model for opinion dynamics in online social networks. In *Proceedings of the ACM Web Conference*, pages 1570–1578, 2022.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations*, 2018.

Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. Nodeaug: Semi-supervised node classification with data augmentation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 207–217, 2020.

Yuelin Wang, Kai Yi, Xinliang Liu, Yu Guang Wang, and Shi Jin. Acmp: Allen-cahn message passing with attractive and repulsive forces for graph neural networks. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pages 6861–6871, 2019.

Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. Continuous graph neural networks. In *International Conference on Machine Learning*, pages 10432–10441. PMLR, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations*, 2019.

Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In Xingquan Zhu, Sanjay Ranka, My T. Thai, Takashi Washio, and Xindong Wu, editors, *IEEE International Conference on Data Mining, 2022*, pages 1287–1292. IEEE, 2022. doi: 10.1109/ICDM54844.2022.00169.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, pages 40–48, 2016.

K. Zhao, Q. Kang, Y. Song, R. She, S. Wang, and W. P. Tay. Graph neural convection-diffusion with heterophily. In *Proc. International Joint Conference on Artificial Intelligence*, Macao, China, Aug 2023.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph neural networks with heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11168–11176, 2021.

Appendix A. Proof

The starting point is to see Eq. 4 as a Riemann sum, here we use high-pass filtering $\widehat{\boldsymbol{L}}_{\boldsymbol{sym}}$, i.e.:

$$\sum_{i=1}^{n+1} \widehat{\boldsymbol{L}}_{\boldsymbol{sym}}{}^{0+(i+1)\dot{\triangle} t} \boldsymbol{H}(0) \triangle t \tag{13}$$

Now letting $n \to \infty$ we get the following integral

$$\boldsymbol{H}_H(t) = \int_0^{t+1} \widehat{\boldsymbol{L}}^{\boldsymbol{s}}_{\boldsymbol{sym}} \boldsymbol{H}(0) ds. \tag{14}$$

The derivative is then given by

$$\frac{\partial \boldsymbol{H}_H(t)}{\partial t} = \widehat{\boldsymbol{L}}^{\boldsymbol{t+1}}_{\boldsymbol{sym}} \boldsymbol{H}(0). \tag{15}$$

For the convenience of solving, we solve the ODE through the second-order ODE and then integrate.

$$\frac{\partial^2 \boldsymbol{H}_H(t)}{\partial t^2} = ln \widehat{\boldsymbol{L}}_{\boldsymbol{sym}} \widehat{\boldsymbol{L}}^{\boldsymbol{t+1}}_{\boldsymbol{sym}} \boldsymbol{H}(0) = ln \widehat{\boldsymbol{L}}_{\boldsymbol{sym}} \frac{\partial \boldsymbol{H}_H(t)}{\partial t}. \tag{16}$$

Now, integrating again

$$\frac{\partial \boldsymbol{H}_H(t)}{\partial t} = ln \widehat{\boldsymbol{L}}_{\boldsymbol{sym}} \boldsymbol{H}(t) + const, \tag{17}$$

and solving for the constant using the fact that

$$\boldsymbol{H}(0) = \int_0^1 \widehat{\boldsymbol{L}}^{\boldsymbol{s}}_{\boldsymbol{sym}} \boldsymbol{H}(0) ds = \frac{\widehat{\boldsymbol{L}}_{\boldsymbol{sym}} - \boldsymbol{I}}{ln \widehat{\boldsymbol{L}}_{\boldsymbol{sym}}} \boldsymbol{H}(0), \tag{18}$$

we get that

$$\frac{\partial \boldsymbol{H}_H(t)}{\partial t}|_{t=0} = \widehat{\boldsymbol{L}}_{\boldsymbol{sym}} \boldsymbol{H}(0) = ln \widehat{\boldsymbol{L}}_{\boldsymbol{sym}} \boldsymbol{H}(0) + const \Rightarrow const = \boldsymbol{H}(0) \tag{19}$$

Therefore, we get the final high-pass filtered ODE

$$\frac{\partial \boldsymbol{H}_H(t)}{\partial t} = ln(\widehat{\boldsymbol{L}}_{\boldsymbol{sym}}) \boldsymbol{H}_H(t) + \boldsymbol{H}(0) = ln(\boldsymbol{I} - \widehat{\boldsymbol{A}}_{\boldsymbol{sym}}) \boldsymbol{H}_H(t) + \boldsymbol{H}(0), \tag{20}$$

in the same way, the ODE of low-pass filtering can be obtained.