Localized Heat Kernel for Graph Neural Networks

Taoyang Qin[1], Ke-Jia Chen[1,2](✉), and Zheng Liu[1,2](✉)

[1] School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, China
[2] Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, Nanjing, China
b21120725@njupt.edu.cn, chenkejia@njupt.edu.cn, zliu@njupt.edu.cn

Abstract. Graph Neural Networks (GNNs) with heat kernel effectively capture the smoothness of labels and features across nodes, preventing oscillations during propagation, and denoising the graph. However, existing models typically employ a global heat kernel, where the diffusion process depends on a single, uniform diffusion time, inevitably resulting in over-smoothing. Additionally, the global heat kernel struggles to handle heterophilic graphs, where nodes exhibit varying neighbor label distributions. To address the above issues, we extend the global heat kernel by a localized scale (i.e., node-level) and integrate it with graph convolution, yielding the Localized Heat Kernel for GNN (LHK-GNN). By adaptively adjusting the diffusion time for each node, our approach enables heat diffusion to accommodate local complexity on graph. Experiments demonstrate the effectiveness of LHK-GNN in mitigating over-smoothing and handling heterophilic graphs.

Keywords: graph convolution · heat kernel · over-smoothing · heterophily.

1 Introduction

Recently, diffusion processes are introduced into Graph Neural Networks (GNNs), achieving superior performance [7,8,27]. Graph heat diffusion is a well-known diffusion process for graph data, which models the information flow across nodes to capture the graph's structural properties. The graph heat kernel [4] is a solution to the heat equation on graphs, which not only encapsulates the temporal evolution of the diffusion process but also serves as a powerful spectral tool. Specifically, many works [27,28] leverage the heat kernel to explore the propagating neighborhood in a continuous manner by tuning the heat diffusion time. Meanwhile, the exponential decay property of the heat kernel suppresses high-frequency signals, thereby enhancing the low-pass filtering effect [8].

However, existing graph heat kernels often employ a single, uniform diffusion time in the diffusion process, which has two major issues. (1) They face the over-smoothing problem, which means they fail to be *deep enough*: as the

number of layers increases, the performance of the GNN degrades significantly [14,12]. From a dynamics perspective, the global heat diffusion process is inherently one where energy gradually dissipates as the diffusion time increases and ultimately converges to a stable thermal equilibrium state [11]. (2) They assume that graphs are homophilic. When handling heterophilic graphs, their performance deteriorates [18,30]. The labels of neighboring nodes vary significantly in heterophilic graphs, resulting in complex local patterns. The global heat kernel relies on a single overall diffusion time, making it ineffective in handling diverse local patterns.

To address the above issues, inspired by the concept of the generalized transfer operator in graph signal processing (GSP) [24] and the design of node-oriented spectral filters [29], we propose the Localized Heat Kernel for Graph Neural Network (LHK-GNN). Specifically, we design an independent and controllable heat diffusion process for each node, allowing the nodes to adaptively adjust the diffusion time. It can prevent excessive information diffusion and control the aggregation scope for relevant node features. Therefore, our method can help mitigate the over-smoothing problem and handle heterophilic graphs. We evaluate LHK-GNN on ten benchmark datasets, and the experimental results validate the effectiveness of our method.

Our contributions are summarized as follows:

- We extend the heat kernel function to the node-level one and integrate it into the spectral graph convolution, resulting in constructing a more fine-grained heat diffusion process.
- We propose a tensor-based method for efficient heat kernel approximation across all nodes. The approach reduces computational demands and demonstrates the successful extension of the heat kernel to the local scale.
- We theoretically prove that the over-smoothing issue observed in existing heat kernel-based GNNs is a consequence of the properties of the heat kernel, explicitly defining the connection.
- Existing experimental results validate the superiority of our method over other diffusion-based GNNs, particularly in mitigating the over-smoothing problem and handling heterophilic graphs. The code is available online[3].

2 Related Work

2.1 Diffusion Process on Graphs

Graph diffusion process is fundamental in graph learning and is typically formulated using partial differential equations (PDEs) to analyze how information diffuses across a graph [16,25,8]. One specific branch of graph diffusion is the heat diffusion model, which characterizes the spread of information in a way similar to heat flowing through connected nodes over time [4,27]. The second branch of graph diffusion is the random walk [25,7], which can be viewed as a

[3] https://github.com/ridethelights/LHKGNN

discrete version of PDE-based diffusion. In a random walk, the state of nodes gradually converges to a stable distribution as time progresses. This process is analogous to the reaching of thermal equilibrium during heat diffusion [3]. The third branch of graph diffusion is anisotropic diffusion, which captures directional information flow by allowing different propagation rates in different regions of the graph [20,6].

2.2 GNNs with Heat Kernel

Xu et al. [27] first combine heat kernel with graph convolution and design a graph convolutional neural network based on heat kernel. Their method effectively captures the smoothness of labels and features across nodes influenced by the graph structure. Gasteiger et al. [8] construct a generalized diffusion matrix that incorporates heat kernel coefficients, and combine it with graph convolution networks, effectively smoothing the neighborhood over the graph and denoising the graph. Zhao et al. [28] combine the heat kernel with Simplifying Graph Convolutional Networks (SGC) [26], which can effectively prevent oscillations. Overall, previous works have guided feature propagation in GNNs from a global heat diffusion perspective. Unlike these methods, our approach assigns an independent heat kernel to each node. Each node can adaptively adjust the heat diffusion range, making the diffusion process more refined.

3 Preliminary

3.1 Notation

Let $\mathcal{G} = (V, E)$ be an undirected graph, where V is the set of nodes with n nodes, and E is the set of edges. Adjacency matrix $\mathbf{A}$ gives the connectivity of $\mathcal{G}$, with $\mathbf{A}_{ij} = \mathbf{A}_{ji}$ denoting the connection between nodes i and j. The normalized graph Laplacian matrix is defined as $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\mathbf{1/2}}\mathbf{A}\mathbf{D}^{-\mathbf{1/2}}$, where $\mathbf{I}$ is the identity matrix, and $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. Since $\mathbf{L}$ is real and symmetric, it has orthonormal eigenvectors $\mathbf{U} = (u_1, u_2, \ldots, u_n)$ with non-negative eigenvalues $\{\lambda_i\}_{i=1}^n$. We assume $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Thus, $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathbf{T}}$, where $\mathbf{\Lambda} = \mathrm{diag}(\{\lambda_i\})$.

3.2 Spectral Graph Convolution

Regarding the eigenvectors of the normalized Laplacian matrix as a set of bases $\mathbf{U}$, the Fourier transform of a graph signal $x \in \mathbb{R}^n$ is defined as $\hat{x} = \mathbf{U}^\top x$, and its inverse is $x = \mathbf{U}\hat{x}$ [24]. Based on the graph Fourier transform, the spectral graph convolution operator $*_\mathcal{G}$ can be defined as:

$$x *_\mathcal{G} \mathbf{g} = \mathbf{U}\left((\mathbf{U}^\top \mathbf{g}) \odot (\mathbf{U}^\top x)\right) = \mathbf{U}\hat{\mathbf{g}}\mathbf{U}x, \tag{1}$$

where $\mathbf{g} \in \mathbb{R}^n$ denotes the kernel in spatial domain, $\hat{\mathbf{g}} = \mathbf{U}^\top \mathbf{g}$ denotes the spectral kernel, and $\odot$ represents Hadamard multiplication.

Early studies typically perform an eigendecomposition on the normalized Laplacian matrix $\mathbf{L}$ to derive the Fourier basis $\mathbf{U}$ and treat the spectral kernel $\hat{\mathbf{g}}$ as the trainable parameters. However, the high computational cost of eigenvalue decomposition severely limits the practical use of these methods. To bypass the eigendecomposition, current works [5,2] usually use the K-order polynomial to approximate different spectral kernels, which can be represented as:

$$\hat{\mathbf{g}} = \hat{g}(\mathbf{\Lambda}) = \sum_{k=0}^{K} \alpha_k \mathbf{\Lambda}^k, \tag{2}$$

where $\mathbf{\Lambda}$ denotes the eigenvalue matrix, $\hat{g}(\cdot)$ denotes the spectral filtering function, and α_k is the learnable coefficient for the k-th order. Inserting into Equation (1), the graph convolution can be represented as:

$$x *_{\mathcal{G}} \mathbf{g} = \mathbf{U} \sum_{k=0}^{K} \alpha_k \mathbf{\Lambda}^k \mathbf{U}^\top x = \sum_{k=0}^{K} \alpha_k \mathbf{L}^k x. \tag{3}$$

3.3 Heat Kernel for Spectral Graph Convolution

The heat kernel is defined as $\mathbf{h}(\lambda_i) = e^{-t\lambda_i}$, where t denotes the diffusion time. According to Xu et al. [27], the convolution based on heat kernel can be represented as:

$$x *_{\mathcal{G}} \mathbf{g} = \mathbf{U} \sum_{k=0}^{K} \alpha_k \mathbf{\Lambda}_t^k \mathbf{U}^\top x, \tag{4}$$

where $\mathbf{\Lambda}_t = \mathrm{diag}\left(\{\mathbf{h}(\lambda_i)\}_{i=1}^n\right)$. In other words, Equation (4) can be represented as: $(\alpha_0\mathbf{I} + \alpha_1 e^{-t\mathbf{L}} + \cdots + \alpha_K e^{-Kt\mathbf{L}})x$. Due to the high computational complexity, K is set to 1, thus $\hat{\mathbf{g}} = \hat{g}(\mathbf{\Lambda}) = \mathrm{diag}\left(\{\alpha_0 + \alpha_1 \mathbf{h}(\lambda_i)\}_{i=1}^n\right)$. When t is small, the computation of $e^{-t\mathbf{L}}$ can be approximated using the Taylor expansion, specifically: $e^{-t\mathbf{L}} \approx \sum_{k=0}^{\infty} \frac{(-t)^k}{k!} \mathbf{L}^k$. A more common approach to approximate $e^{-t\mathbf{L}}$ is using Chebyshev polynomials [10].

4 Methodology

Inspired by the generalized transition operator in graph signal processing [24] and the design of node-oriented spectral filters [29], we design the Localized Heat Kernel for GNN (LHK-GNN) that effectively overcomes the issues of over-smoothing and heterophily.

4.1 Spectral Graph Convolution with Localized Heat Kernel

We first adapt the spectral graph convolution to operate on individual nodes. Then we extend the graph heat kernel to the node level, creating a localized heat kernel. The localized kernel is then integrated with node-oriented spectral graph convolution. The design of our approach is based on the generalized transition operator [24], defined as follows:

Definition 1. *For any spatial kernel* $\mathbf{g} \in \mathbb{R}^n$ *on a given graph* $\mathcal{G}$*, and any* $i \in \{1, \dots n\}$*, the generalized transition operator* $\mathbf{T}_i : \mathbb{R}^n \to \mathbb{R}^n$ *is expressed via generalized convolution with the Kronecker delta function* δ_i *centered at node* i*:*

$$\mathbf{T}_i(\mathbf{g}) := \sqrt{N}(\mathbf{g} * \delta_i) = \sqrt{N} \sum_{l=1}^{n} u_l u_l^\top(i) \hat{g}(\lambda_l).$$

.

For node-oriented filtering [29], $\mathbf{g}$ is first aligned at node i using the operator $\mathbf{T}_i$, then spectrally convolved with x. This operation can be written as:

$$x *_{\mathcal{G}} \mathbf{T}_i(\mathbf{g}) = \sqrt{N} \sum_{l=1}^{n} u_l \hat{x}(\lambda_l) u_l^T(i) \hat{g}(\lambda_l). \tag{5}$$

Defining $\hat{g}_i(\lambda_l)$ as: $\hat{g}_i(\lambda_l) = \sqrt{N} u_l^T(i) \hat{g}(\lambda_l)$, which leads to the following expression for the convolution:

$$x *_{\mathcal{G}} \mathbf{T}_i(\mathbf{g}) = \sum_{l=1}^{n} \hat{g}_i(\lambda_l) u_l u_l^T x = \mathbf{U} \hat{\mathbf{g}}_i \mathbf{U}^T x. \tag{6}$$

Here, $\hat{g}_i(\lambda_l)$ represents the value of the signal at node i after the spectral filtering operation, where $u_l^T(i)$ is the i-th element of the l-th eigenvector u_l, and $\hat{g}(\lambda_l)$ is the spectral filter evaluated at eigenvalue λ_l. Considering the $\hat{g}(\cdot)$ in Section 3.3, the $\hat{g}_i(\lambda_l)$ can be represented as:

$$\begin{aligned} \hat{g}_i(\lambda_l) &= \sqrt{N} u_l^T(i) \left(\alpha_0 + \alpha_1 e^{-t\lambda_l}\right) \\ &= \sqrt{N} u_l^T(i) \alpha_0 + \sqrt{N} u_l^T(i) \alpha_1 e^{-t\lambda_l}. \end{aligned} \tag{7}$$

Now, we extend the heat kernel in Section 3.3 to the node level by using a localized scale: $\mathbf{h}_i(\lambda_l) = e^{-t_i \lambda_l}$, which serves as the heat kernel for node i. Due to the learnability of α_0 and α_1, we can approximate $\sqrt{N} u_l^T(i) \alpha_0$ and $\sqrt{N} u_l^T(i) \alpha_1 e^{-t\lambda_l}$ as γ_{i0} and $\gamma_{i1} \mathbf{h}_i(\lambda_l)$, respectively. Based on this, we have the following:

$$\begin{aligned} \hat{g}_i(\lambda_l) &\approx \gamma_{i0} + \gamma_{i1} \mathbf{h}_i(\lambda_l), \\ \hat{\mathbf{g}}_i &= \operatorname{diag}\left(\{\gamma_{i0} + \gamma_{i1} \mathbf{h}_i(\lambda_l)\}_{l=1}^n\right). \end{aligned} \tag{8}$$

Thus, the convolution with localized (i.e., node-level) heat kernel can be represented as follows:

$$\begin{aligned} x *_{\mathcal{G}} \mathbf{T}_i(\mathbf{g}) &= \delta_i \left(\mathbf{U} \left(\gamma_{i0} \mathbf{I} + \gamma_{i1} e^{-t_i \mathbf{\Lambda}}\right) \mathbf{U}^\top x\right) \\ &= \delta_i (\gamma_{i0} \mathbf{I} + \gamma_{i1} e^{-t_i \mathbf{L}}) x, \end{aligned} \tag{9}$$

where $\delta_i = [0, 0, \dots, 1, \dots, 0]$ denotes a row vector with the i-th element being 1 and the remaining elements being zeros.

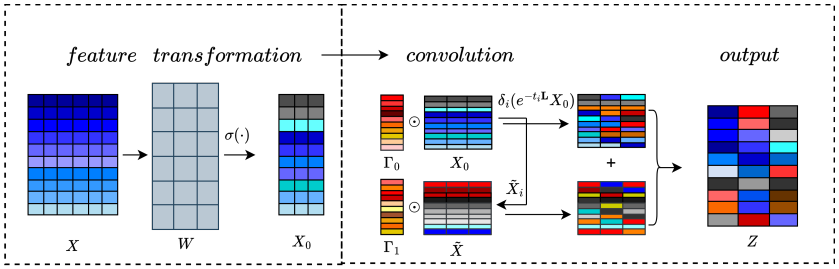


Fig. 1: The workflow. The feature matrix X is first transformed by an MLP to obtain X_0. $A \odot B$ denotes the Hadamard product, which applies broadcasting to A if necessary. After undergoing localized heat kernel convolution, we obtain $\tilde{X}$, where $\tilde{X}_i = \delta_i(e^{-t_i\mathbf{L}}X_0)$. Finally, the output Z is a weighted combination of X_0 and $\tilde{X}$.

4.2 The Implementation of LHK-GNN

This section describes the implementation of LHK-GNN, including its overall architecture and the method used to approximate the localized heat kernel. Figure 1 presents the overall workflow.

Architecture. To reduce the dimensionality of features and enhance the model's performance, we firstly employ an MLP to non-linearly transform [9] the raw feature matrix $X \in \mathbb{R}^{n\times f}$, yielding $X_0 = \mathrm{MLP}(X;\Theta)$. Thus the feature propagation rule of LHK-GNN can be represented as:

$$Y_i = \delta_i(\gamma_{i0}\mathbf{I} + \gamma_{i1}e^{-t_i\mathbf{L}})X_0. \tag{10}$$

We treat t_i as a learnable parameter, enabling each node to adaptively adjust the diffusion time. For the node classification task, we adopt the negative log-likelihood (NLL) loss. Thus we need to apply the softmax function [1] on Y_i to obtain the predication $Z_i = softmax(Y_i)$.

Approximation for Localized Heat Kernel. We use the Chebyshev polynomial to approximate $e^{-t_i\mathbf{L}}$. Specifically,

$$e^{-t_i\mathbf{L}} \approx \sum_{k=0}^{K-1} c_k(t_i)T_k(\tilde{\mathbf{L}}),$$

where $\tilde{\mathbf{L}} = \mathbf{L} - \mathbf{I}$ and $T_k(\tilde{\mathbf{L}})$ denotes the k-th Chebyshev polynomial. The coefficients $c_k(t_i)$ depend on the diffusion time t_i and can be computed as follows:

$$c_k(t_i) = \frac{\beta}{\pi}\int_{-1}^{1} T_k(x)e^{-t_i(1+x)}\,dx = \beta(-1)^k e^{-t_i}I_k(t_i), \tag{11}$$

Algorithm 1: LHK-GNN

Input: Feature matrix X, scaled Laplacian $\tilde{L}$, number of expansions K
Parameters: Learnable vector $\mathcal{T}$, parameters Γ_0, Γ_1, MLP parameters Θ
Note: $I_k(\mathcal{T})$ denotes the modified Bessel function of the first kind; $A \odot B$ denotes the Hadamard product, which applies broadcasting to A if necessary.

Feature Transformation:
$X_0 \leftarrow \mathrm{MLP}(X; \Theta)$;
Chebyshev Expansion:
for $k \leftarrow 0$ **to** $K-1$ **do**
  if $k = 0$ **then**
    $c_0 \leftarrow \exp(-\mathcal{T}) \odot I_0(\mathcal{T})$;
  else
    $c_k \leftarrow 2\exp(-\mathcal{T}) \odot (-1)^k I_k(\mathcal{T})$;
  end
end
Chebyshev Polynomials:
$T_0 \leftarrow X$;
$T_1 \leftarrow \tilde{\mathbf{L}} X$;
for $k \leftarrow 2$ **to** $K-1$ **do**
  $T_k \leftarrow 2\tilde{\mathbf{L}} T_{k-1} - T_{k-2}$;
end
$\tilde{X} \leftarrow \sum_{k=0}^{K} (c_k \odot T_k)$;
Output:
$Z \leftarrow \boldsymbol{\Gamma}_0 \odot X_0 + \boldsymbol{\Gamma}_1 \odot \tilde{X}$;

where $\beta = 1$ for $k = 0$ and $\beta = 2$ for $k \geq 1$, and $I_k(t_i)$ is the modified Bessel function of the first kind. By combining the revised coefficients, we obtain the following approximation for $e^{-t_i L}$:

$$e^{-t_i \mathbf{L}} \approx e^{-t_i} I_0(t_i) T_0(\tilde{\mathbf{L}}) + 2 \sum_{k=1}^{K-1} (-1)^k e^{-t_i} I_k(t_i) T_k(\tilde{\mathbf{L}}). \tag{12}$$

In practice, implementing LHK-GNN by iteratively approximating each node's heat kernel would require an enormous amount of computational resources. Therefore, a more efficient approach is to approximate the heat kernels for all nodes simultaneously using a tensor-based method (as shown in Algorithm 1), which avoids the overhead associated with iterative traversal.

4.3 Analysis for LHK-GNN: Tackling Over-Smoothing and Heterophily

Compared to other GNNs with heat kernel, the most significant distinction of our model is its ability to adaptively adjust the heat diffusion time for each node, which helps address the issues of over-smoothing and heterophily.

Tackling Over-Smoothing. Over-smoothing refers to the phenomenon in GNNs where node representations become indistinguishable from each other as the number of layers increases. Consequently, the performance of GNNs gradually deteriorates with deeper architectures. Based on Chien et al. [2], we formally define the over-smoothing problem as follows:

Definition 2. *Given a propagation matrix $\mathcal{A}$ and a graph feature matrix X, the k-step feature propagation can be expressed as $X^{(k)} = \mathcal{A}^k X^{(0)}$. If $\lim_{k\to\infty} X^{(k)} = X^{(\infty)}$, the phenomenon is referred to as over-smoothing.*

According to Definition 2, the over-smoothing problem can be equivalently described as $\lim_{k\to\infty} \mathcal{A}^k = \mathcal{A}^\infty$, meaning that as the power k increases, $\mathcal{A}$ converges to a steady-state matrix. Based on the above discussion, we theoretically analyze that traditional heat kernel-based GNNs may exhibit the over-smoothing phenomenon. Specifically, we have the following:

Proposition 1. *Let $\mathcal{A} = \alpha_0\mathbf{I} + \alpha_1 e^{-t\mathbf{L}}$, where $\mathbf{L}$ is the normalized Laplacian, $\alpha_0 + \alpha_1 = 1$, and $0 < \alpha_1 < 2$. Then, for any $t > 0$, the sequence of matrix powers converges, i.e., $\lim_{k\to\infty} \mathcal{A}^k = \mathcal{A}^\infty$, where $\mathcal{A}^\infty$ is a projection matrix onto the invariant subspace corresponding to the eigenvalue 1.*

Proof. Since $\mathbf{L}$ is the normalized Laplacian, it is symmetric and diagonalizable. Let $\{\lambda_i\}$ be its eigenvalues (with $\lambda_i \geq 0$), and denote by $\mathbf{U}$ an orthogonal matrix that diagonalizes $\mathbf{L}$.

Then, the heat kernel can be written as $e^{-t\mathbf{L}} = \mathbf{U}e^{-t\mathbf{\Lambda}}\mathbf{U}^T$, where $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues. Consequently, $e^{-t\mathbf{L}}$ is also symmetric and diagonalizable with eigenvalues $e^{-t\lambda_i}$. Thus, the eigenvalues of $\mathcal{A} = \alpha_0\mathbf{I} + \alpha_1 e^{-t\mathbf{L}}$ are given by $\mu_i = \alpha_0 + \alpha_1 e^{-t\lambda_i}$. In particular, when $\lambda_i = 0$, we have

$$\mu_i = \alpha_0 + \alpha_1 = 1. \tag{13}$$

For $\lambda_i > 0$, note that

$$\mu_i = 1 - \alpha_1 \left(1 - e^{-t\lambda_i}\right). \tag{14}$$

Since $0 < e^{-t\lambda_i} < 1$ for $t > 0$ and $\lambda_i > 0$, and given $0 < \alpha_1 < 2$, it follows that $|\mu_i| < 1$. Because $\mathcal{A}$ is diagonalizable (and hence all eigenvalues are semisimple), taking powers of $\mathcal{A}$ yields $\mathcal{A}^k = \mathbf{U}(\theta_0\mathbf{I} + \theta_1 e^{-t\mathbf{\Lambda}})^k\mathbf{U}^T$. As $k \to \infty$, all components corresponding to eigenvalues with $|\mu_i| < 1$ decay to zero, while the components corresponding to $\mu_i = 1$ remain unchanged. Thus,

$$\lim_{k\to\infty} \mathcal{A}^k = \mathcal{A}^\infty, \tag{15}$$

where $\mathcal{A}^\infty$ is the projection onto the invariant subspace associated with the eigenvalue 1.

This completes the proof.

The above analysis of over-smoothing in heat kernels is based on feature iteration. From a diffusion perspective, we can similarly conclude that heat kernels

induce over-smoothing. Considering the heat kernel e^{-tL}, as $t \to \infty$, it converges to a steady-state matrix. This indicates that when the heat diffusion time is too long, the graph approaches a *thermal equilibrium* state, where node *energy* no longer changes with increasing diffusion time. Therefore, reducing t can alleviate the over-smoothing issue.

However, an excessively small t may compromise the properties of the heat kernel, preventing sufficient smoothing of graph labels. Thus, adaptively adjusting the heat diffusion time t for each node provides a more flexible way to control the diffusion process. According to [13], nodes with high degrees tend to converge to their stable states earlier than low-degree nodes, as evident. Based on this, for nodes with high degrees and similar or identical neighbor labels, a smaller t is required to prevent rapid convergence to a *thermal equilibrium* state. Conversely, for nodes with diverse or highly dissimilar neighbor labels, a larger t is needed. This allows for identifying similar nodes by adjusting the diffusion range while preventing the incorporation of misleading information due to an overly restricted scope.

Tackling Heterophily. Heterophily refers to the tendency of connected nodes to have different labels or features. According to [19], the node-level homophily ratio is a metric for measuring node homophily, defined as follows:

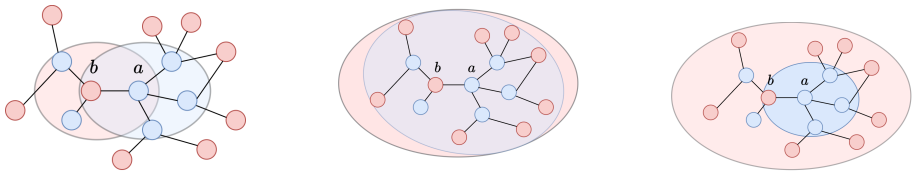
Definition 3. *Let $G = (V, E)$ be a labeled graph where each node v has a label $y(v)$. The node-level homophily of a node v is defined as:*

$$\mathcal{H}(v) = \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \mathbf{1}\big[y(u) = y(v)\big],$$

where $\mathcal{N}(v)$ denotes the set of neighbors of v, and $\mathbf{1}[\cdot]$ is an indicator function that returns 1 if the condition is true, and 0 otherwise.

Traditional models rely on a global heat diffusion time to process graph data, which performs well on homophilic graphs but often proves ineffective on heterophilic graphs. The reason is that these methods are fundamentally based on the homophily assumption, which posits that connected nodes tend to have the same label or similar features. From the perspective of heat diffusion, since the diffusion time t defines the range of neighbors [27], traditional methods assign the same neighbor range to each node, which does not guarantee the effective exclusion of nodes with different labels for every node. In contrast, the node-specific LHK-GNN defines an independent neighbor range for each node (as shown in Figure 2), which theoretically allows for the effective exclusion of nodes with different labels for each node.

Balancing Over-Smoothing and Heterophily. A special case arises when high-degree nodes exhibit low homophily. If a small diffusion time is applied, it may fail to effectively aggregate features from similar nodes, leading to performance degradation. To address this, one solution is to categorize nodes into



(a) The optimal diffusion range of node a

(b) The optimal diffusion range of node b

(c) The diffusion range of LHK-GNN

Fig. 2: Illustration of different diffusion ranges. (a): We restrict diffusion to integer-order neighbors. Since most of node a's first-order neighbors share its label, its optimal range is first-order (light blue). However, in this setting, node b aggregates features only from its first-order neighbors, whose labels differ from b, preventing it from effectively capturing useful information. (b): Node b's optimal range is third-order (light pink). In this setting, node a fails to learn effective features. (c): Traditional methods enforce a uniform range like (a) and (b), preventing optimal diffusion. LHK-GNN instead enables each node to diffuse within its optimal range.

different intervals based on their homophily levels, measured by homophily ratio in Definition 3, and assign each interval an upper bound ϵ_i and a lower bound η_i for diffusion time. The diffusion time for each node is then learned within these threshold constraints. The upper bound in high-degree intervals is set lower than that in low-degree intervals, though the difference remains moderate to ensure that high-degree nodes can still adaptively adjust their diffusion range according to the downstream task's loss function. This approach prevents excessively small diffusion times from failing to aggregate meaningful features while avoiding overly large diffusion times that lead to over-smoothing, thereby achieving a balance between over-smoothing and heterophily.

5 Experiments

In this section, we evaluate the performance of LHK-GNN on node classification tasks across 10 benchmark datasets. Section 5.1 provides details on the datasets, baselines, and hyperparameter settings used in the experiments. Section 5.2 presents the experimental results, including the performance on both homophilic and heterophilic graphs. Section 5.3 provides an analysis of the influence of diffusion time. Section 5.4 provides the parameter study.

5.1 Experimental Setup

Datasets. We evaluate our model on ten commonly used real-world datasets. Among them, five are homophilic datasets: Cora, CiteSeer, and PubMed [22], Computers, and Photo [23], and the other are heterophilic datasets: Chameleon

Table 1: Statistics of the datasets

	Homophilic datasets					Heterophilic datasets				
Datasets	Cora	Cite.	PubMed	Comp.	Photo	Cham.	Squi.	Actor	Texas	Corn
Nodes	2,708	3,327	19,717	13,752	7,650	2,277	5,201	7,600	183	183
Edges	5,278	4,552	44,324	245,861	119,081	31,371	198,353	26,659	279	277
Features	1,433	3,703	500	767	745	2,325	2,089	932	1,703	1,703
Classes	7	6	3	10	8	5	5	5	5	5
$\mathcal{H}_\mathcal{G}$	0.81	0.70	0.79	0.80	0.83	0.24	0.22	0.21	0.05	0.30

and Squirrel [21], Actor, Texas, and Cornell [19]. Besides, we follow [2] to adopt two data splitting strategies: the sparse splitting 2.5%/2.5%/95% for training, validation, and testing, respectively, and the dense splitting 60%/20%/20%. For experiments, we perform 100 runs with different random splits. The dataset statistics are provided in Table 1. Note that $H_\mathcal{G}$ is used to measure the proportion of homophily in the graph, defined as: $\mathcal{H}_\mathcal{G} = \frac{|\{(u,v)\in E|y_u=y_v\}|}{|E|}$.

Baselines. We select commonly used diffusion-based GNNs as baselines, including GCN [15], GraphHeat [27], APPNP [7], GDC [8], HKGCN [28], GPRGNN [2], and HiD-Net [17]. Among them, GCN, GraphHeat, and HKGCN can be regarded as heat kernel-based GNNs (with GCN considered as a first-order heat diffusion GNN). APPNP and GPRGNN are random walk-based GNNs, where their iterative node feature updates can still be viewed as an information diffusion process. GDC defines a generalized diffusion matrix that supports multiple diffusion modes, including heat diffusion. HiD-Net constructs a generalized neural diffusion framework on Graphs.

Hyper-Parameters. For the baselines, we use the best combination of hyperparameters provided in the original paper to report the results for each dataset. For our proposed LHK-GNN, we use a 2-layer MLP for feature transformation. For each dataset, we search for the optimal hyperparameters within {1e-3, 5e-3, 1e-2, 5e-2} for learning rate, within {0.3, 0.5, 0.7, 0.8, 0.9} for dropout, within {16, 32, 64} for hidden channels, and within {0, 1e-5, 5e-5, 1e-4, 5e-4} for weight decay to achieve the best performance. In all experiments, we use the Adam optimizer and apply standard early stopping after 200 epochs.

5.2 Experimental Results

We use sparse splitting for homophilic graphs and dense splitting for heterophilic graphs according to [2]. The results are presented in Table 2 and Table 3, respectively. The results show that LHK-GNN achieves the best performance on 7 datasets and the second-best performance on 3 datasets. Compared to the

Table 2: Results on real-world homophilic datasets in the sparse splitting (2.5%/2.5%/95%): Mean accuracy across runs (%) ± 95% confidence interval. Boldface denotes the best results, and underlining denotes the second-best.

Method	Cora	CiteSeer	PubMed	Computers	Photo
GCN	76.43±0.87	66.71±0.73	84.35±0.90	82.46±0.75	90.61±1.08
GraphHeat	77.67±1.04	66.84±0.96	83.62±1.12	82.78±0.63	90.68±0.89
APPNP	78.74±1.06	67.34±1.02	84.46±0.83	82.13±0.67	90.43±1.15
GDC	77.79±0.78	67.16±0.55	<u>84.62</u>±0.68	82.78±0.72	91.19±1.10
GPRGNN	79.34±0.81	<u>67.73</u>±0.72	84.27±1.14	82.97±0.90	<u>91.72</u>±0.95
HKGCN	78.52±0.92	67.05±0.91	83.89±0.88	82.34±0.59	90.75±0.97
HiD-Net	<u>79.53</u>±1.03	66.87±0.99	84.32±0.74	**84.14**±0.60	91.45±0.91
LHK-GNN	**80.02**±1.13	**67.92**±0.87	**84.81**±0.56	<u>83.36</u>±0.75	**92.3**±1.25

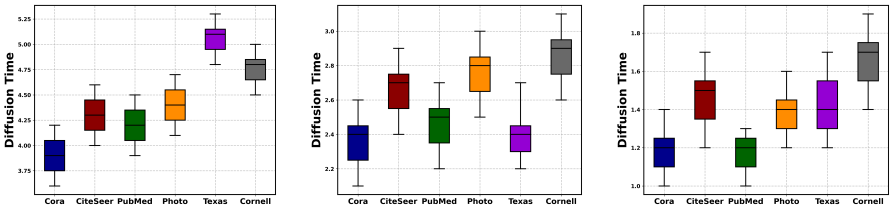
Table 3: Results on real-world heterophilic datasets in the dense splitting (60%/20%/20%): Mean accuracy across runs (%) ± 95% confidence interval. Boldface denotes the best results, and underlining denotes the second-best.

Method	Chameleon	Squirrel	Actor	Texas	Cornell
GCN	61.21 ±0.95	44.92 ±0.87	32.14 ±0.94	76.45 ±1.20	68.31 ±0.88
GraphHeat	64.68 ±0.79	45.61 ±0.66	34.51 ±1.02	80.82 ±1.08	72.92 ±1.15
APPNP	62.16 ±0.91	40.66 ±0.75	39.44 ±1.11	91.24 ±0.87	91.26 ±1.00
GDC	67.53 ±0.92	44.83 ±0.81	39.32 ±1.02	91.66 ±1.05	91.43 ±0.98
GPRGNN	67.78 ±0.82	<u>50.27</u> ±0.65	39.65 ±0.85	91.83 ±0.91	92.17 ±0.88
HKGCN	63.44 ±0.88	46.76 ±0.93	34.73 ±1.06	78.63 ±0.82	74.66 ±1.04
HiD-Net	<u>68.41</u> ±0.81	48.86 ±0.79	**40.07** ±0.93	<u>92.54</u> ±0.95	**92.37** ±1.10
LHK-GNN	**70.67** ±0.92	**56.34** ±0.87	<u>39.88</u> ±0.95	**93.47** ±1.08	<u>92.24</u> ±1.20

baselines, LHK-GNN has superior performance on heterophilic graph datasets, especially on Chameleon and Squirrel, LHK-GNN outperforming the baselines by an average of 5.64% and 10.35%, respectively. Apart from that, compared to heat kernel-based GNNs such as GraphHeat and HKGCN, LHK-GNN achieves significant performance gains on every dataset——including an average improvement of about 12% on heterophilic graphs. These comparisons demonstrate that LHK-GNN is more effective than existing diffusion-based GNNs in handling both homophilic and heterophilic graphs.

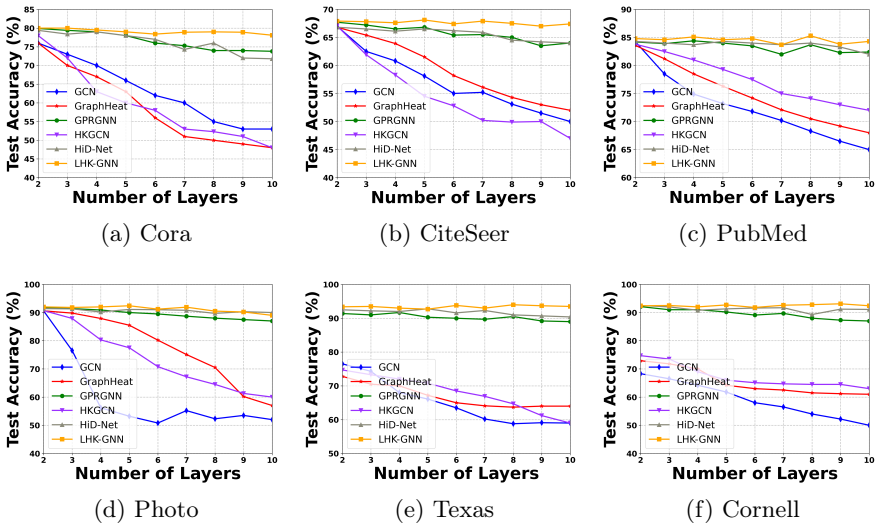
5.3 The Influence of Diffusion Time

Each node has an independent heat diffusion time, resulting in a complex heat diffusion pattern. To better understand how this heat diffusion pattern helps mitigate over-smoothing and handle heterophilic graph datasets, we study the



(a) I1 (b) I2 (c) I3

Fig. 3: Boxplots of diffusion times for each dataset across three intervals.



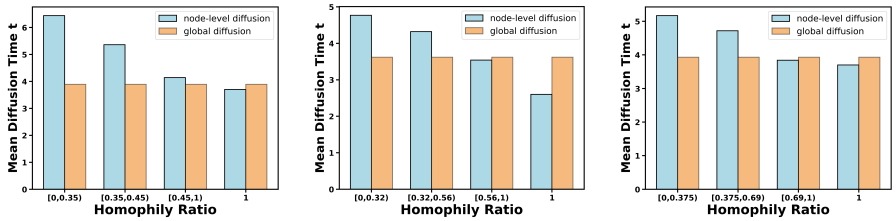
(a) Cora (b) CiteSeer (c) PubMed

(d) Photo (e) Texas (f) Cornell

Fig. 4: Test accuracy with increasing number of propagation steps.

relationship between each node's diffusion time and its degree, and separately, between its diffusion time and its heterophily ratio.

Over-Smoothing. The key to mitigating over-smoothing is to adjust the diffusion time for different local patterns. According to Section 4.3, different local patterns can be distinguished based on the degree of the nodes. Taking 6 datasets as examples, we divide the diffusion time into five intervals based on the degree and train the diffusion time in each interval I_i , constraining it with different thresholds ϵ_i and η_i . The results, as shown in Figure 3. Under the settings specified in Figure 3 for LHK-GNN, we compare LHK-GNN with other graph diffusion models while varying the propagation step k from 2 to 10. The baselines include two models that are effective in handling over-smoothing—GPRGNN and HiD-Net—as well as other models, such as GCN, GraphHeat, and HKGCN.



(a) Chameleon (b) Squirrel (c) Texas

Fig. 5: LHK-GNN and GraphHeat diffusion time comparison.

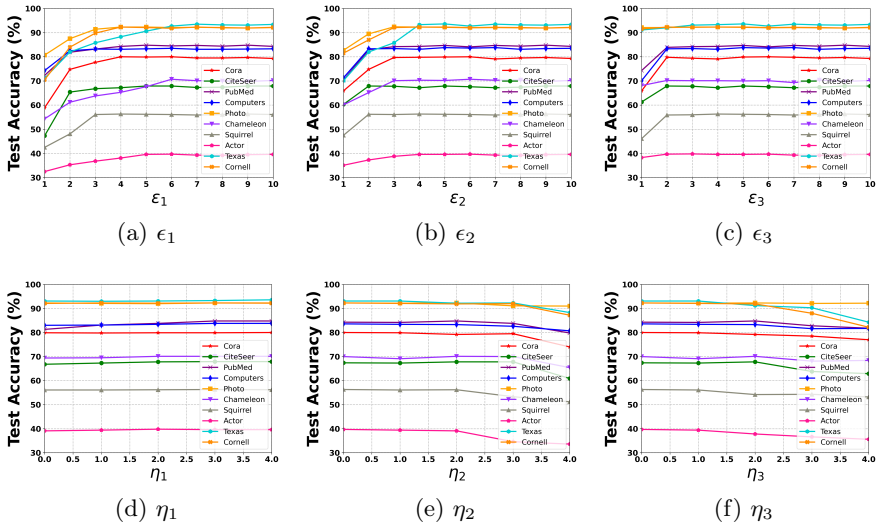
The results are shown in Figure 4. From the results, we can observe that with the increase of k, LHK-GNN consistently performs better than other baselines.

Heterophily. According to section 4.3, the key to handling heterophilic datasets is adaptively adjusting the heat diffusion time for each node based on the level of homophily ratio. To verify whether our model can adjust the heat diffusion range based on nodes with different homophily ratios, we relate the heat diffusion times learned by LHK-GNN and GraphHeat to the homophily ratio of each node. Using Chameleon, Squirrel and Texas as examples, the results are shown in Figure 5. The results show that LHK-GNN increases diffusion time in low-homophily regions and decreases it in high-homophily regions, while GraphHeat keeps the diffusion time unchanged. This validates that our method can adaptively adjust diffusion time based on the homophily of nodes, thereby enabling flexible and effective handling of heterophilic graphs.

5.4 Parameter Study

In this section, we investigate the sensitivity of parameters on all datasets. There are two types of parameters: each interval has an upper bound ϵ_i and a lower bound η_i.

From the results (as shown in Figure 6), we have following observations: (1) as ϵ_i increases, the model's performance on all datasets improves and converges to a stable state. This implies that when ϵ_i is sufficiently large, the model can learn the optimal diffusion time in each interval and is not affected by further increases in ϵ_i. (2) Increasing η_1 has no impact on performance, indicating that low-degree nodes (interval I_1) require a longer diffusion time for effective feature aggregation. In contrast, rising η_2 and η_3 gradually degrade performance, suggesting that nodes in intervals I_2 and I_3 (with higher degrees) need a moderate diffusion time to avoid aggregating too many harmful features.



(a) ϵ_1 (b) ϵ_2 (c) ϵ_3

(d) η_1 (e) η_2 (f) η_3

Fig. 6: Parameter influence on performance.

6 Conclusion

In this paper, we extend the heat kernel to the node level and integrate it with graph convolution to propose the Localized Heat Kernel for GNN (LHK-GNN). Our method allows each node to adaptively adjust the diffusion time, enabling different diffusion ranges based on node degree and the similarity of neighboring labels. This approach theoretically alleviates over-smoothing and effectively handles heterophilic graphs. Experimental results demonstrate the effectiveness of our method in mitigating over-smoothing and handling heterophilic graphs.

Acknowledgments. We would like to thank reviewers for their constructive comments.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bridle, J.S.: Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In: NATO Neurocomputing (1989), `https://api.semanticscholar.org/CorpusID:59636530`
2. Chien, E., Peng, J., Li, P., Milenkovic, O.: Adaptive universal generalized pagerank graph neural network. arXiv preprint arXiv:2006.07988 (2020)
3. Chung, F.: The heat kernel as the pagerank of a graph. Proceedings of the National Academy of Sciences **104**(50), 19735–19740 (2007)

4. Chung, F.R.: Spectral graph theory, vol. 92. American Mathematical Soc. (1997)
5. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems **29** (2016)
6. Fu, G., Zhao, P., Bian, Y.: p p -laplacian based graph neural networks. In: International conference on machine learning. pp. 6878–6917. PMLR (2022)
7. Gasteiger, J., Bojchevski, A., Günnemann, S.: Predict then propagate: Graph neural networks meet personalized pagerank. arXiv preprint arXiv:1810.05997 (2018)
8. Gasteiger, J., Weißenberger, S., Günnemann, S.: Diffusion improves graph learning. Advances in neural information processing systems **32** (2019)
9. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 315–323. JMLR Workshop and Conference Proceedings (2011)
10. Hammond, D.K., Vandergheynst, P., Gribonval, R.: Wavelets on graphs via spectral graph theory. Applied and Computational Harmonic Analysis **30**(2), 129–150 (2011)
11. Han, A., Shi, D., Lin, L., Gao, J.: From continuous dynamics to graph neural networks: Neural diffusion and beyond. arXiv preprint arXiv:2310.10121 (2023)
12. Hossain, T., Saifuddin, K.M., Islam, M.I.K., Tanvir, F., Akbas, E.: Tackling oversmoothing in gnn via graph sparsification: A truss-based approach. arXiv preprint arXiv:2407.11928 (2024)
13. Huang, R., Li, P.: Hub-hub connections matter: Improving edge dropout to relieve over-smoothing in graph neural networks. Knowledge-Based Systems **270**, 110556 (2023)
14. Keriven, N.: Not too little, not too much: a theoretical analysis of graph (over) smoothing. Advances in Neural Information Processing Systems **35**, 2268–2281 (2022)
15. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
16. Kondor, R.I., Lafferty, J.: Diffusion kernels on graphs and other discrete structures. In: Proceedings of the 19th international conference on machine learning. vol. 2002, pp. 315–322 (2002)
17. Li, Y., Wang, X., Liu, H., Shi, C.: A generalized neural diffusion framework on graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 8707–8715 (2024)
18. Luan, S., Hua, C., Lu, Q., Zhu, J., Zhao, M., Zhang, S., Chang, X.W., Precup, D.: Is heterophily a real nightmare for graph neural networks to do node classification? arXiv preprint arXiv:2109.05641 (2021)
19. Pei, H., Wei, B., Chang, K.C.C., Lei, Y., Yang, B.: Geom-gcn: Geometric graph convolutional networks. arXiv preprint arXiv:2002.05287 (2020)
20. Poli, M., Massaroli, S., Park, J., Yamashita, A., Asama, H., Park, J.: Graph neural ordinary differential equations. arXiv preprint arXiv:1911.07532 (2019)
21. Rozemberczki, B., Allen, C., Sarkar, R.: Multi-scale attributed node embedding. Journal of Complex Networks **9**(2), cnab014 (2021)
22. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI magazine **29**(3), 93–93 (2008)
23. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868 (2018)
24. Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A., Vandergheynst, P.: The emerging field of signal processing on graphs: Extending high-dimensional data

analysis to networks and other irregular domains. IEEE signal processing magazine **30**(3), 83–98 (2013)
25. Tsiatas, A.: Pagerank and diffusion on large graphs. UCSD Research Exam **14** (2009)
26. Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K.: Simplifying graph convolutional networks. In: International conference on machine learning. pp. 6861–6871. Pmlr (2019)
27. Xu, B., Shen, H., Cao, Q., Cen, K., Cheng, X.: Graph convolutional networks using heat kernel for semi-supervised learning. arXiv preprint arXiv:2007.16002 (2020)
28. Zhao, J., Dong, Y., Tang, J., Ding, M., Wang, K.: Generalizing graph convolutional networks via heat kernel (2021)
29. Zheng, S., Zhu, Z., Liu, Z., Li, Y., Zhao, Y.: Node-oriented spectral filtering for graph neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(1), 388–402 (2023)
30. Zheng, X., Wang, Y., Liu, Y., Li, M., Zhang, M., Jin, D., Yu, P.S., Pan, S.: Graph neural networks for graphs with heterophily: A survey. arXiv preprint arXiv:2202.07082 (2022)