Implicit Graph Neural Diffusion Based on Constrained Dirichlet Energy Minimization

Guoji Fu **Mohammed Haroon Dupty** **Yanfei Dong** **Lee Wee Sun**
School of Computing
National University of Singapore
{guojifu, dmharoon, dyanfei, leews}@comp.nus.edu.sg

Abstract

Implicit graph neural networks (GNNs) have emerged as a potential approach to enable GNNs to capture long-range dependencies effectively. However, poorly designed implicit GNN layers can experience over-smoothing or may have limited adaptability to learn the graph geometry, potentially hindering their performance in graph learning problems. To address these issues, we introduce a geometric framework to design implicit graph diffusion layers based on a parameterized graph Laplacian operator. Our framework allows learning the metrics of vertex and edge spaces, as well as the graph gradient operator from data. We further show how implicit GNN layers can be viewed as the fixed-point solution of a Dirichlet energy minimization problem and give conditions under which it may suffer from over-smoothing. To overcome the over-smoothing problem, we design our implicit graph diffusion layer as the solution of a Dirichlet energy minimization problem with constraints on vertex features, enabling it to trade off smoothing with the preservation of node feature information. With an appropriate hyperparameter set to be larger than the largest eigenvalue of the parameterized graph Laplacian, our framework guarantees a unique equilibrium and quick convergence. Our models demonstrate better performance than leading implicit and explicit GNNs on benchmark datasets for node and graph classification tasks, with substantial accuracy improvements observed for some datasets.

1 Introduction

Graph neural networks (GNNs) have been shown to have superior representation learning ability. Existing GNN architectures, including graph convolutional networks [1, 2, 3, 4], graph attention networks [5, 6], and message passing neural networks [7, 8, 9, 10], have achieved great success in many applications in social media [11], recommender systems [12, 13], natural language processing [14, 15], and computational biology [16, 7, 17]. However, despite the successes, conventional GNNs typically lack the ability to capture long-range dependencies [18]. GNN layers update the node representation by aggregating the information with T propagation steps, hence each node can only gather information from its T-hop neighbors. Moreover, the representations of all nodes will become indistinguishable as the propagation steps increase, leading to a significant decline in the performance of GNNs, a phenomenon commonly referred to as over-smoothing [18, 19], which limits the capacity of GNNs to capture global information.

One potential way to tackle the problem is to use an implicit layer [20, 21] to replace deeply stacked explicit propagation layers. The implicit layer solves a fixed point equilibrium equation based on explicit propagation schemes and is equivalent to running infinite propagation steps [22, 23, 24, 25]. By solving the fixed point equation using root-finding methods, the obtained equilibrium can then be used as node representations, potentially containing information from infinite-hop neighbors.

NeurIPS 2023 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2023).

However, the performance of implicit GNNs depends strongly on the adopted implicit layer. Poorly designed implicit GNNs may suffer from over-smoothing, limiting them to benefit from long-range dependencies. Specifically, simply generalizing recurrent GNN layers to implicit layers, such as [22, 26], could inherit the over-smoothing issue [27, 28, 29, 24]. Moreover, the underlying geometry of the graph has been shown to have a profound connection with the performance of GNNs in heterophilic settings and their over-smoothing behaviours [30]. However, existing implicit GNNs, e.g., [31, 22, 23, 25, 24, 26, 29], have primarily concentrated on learning the diffusion strength without explicitly engaging in learning the geometry of vertex and edge spaces, which may limit their adaptability to learn the graph geometry and potentially hinder their performance in graph learning problems. Furthermore, reliable convergence of implicit layers is another desirable criterion [32].

To confront the above challenges, we propose a geometric framework to design implicit graph neural diffusion layers based on a *parameterized graph Laplacian operator*. We first define a Hilbert space for vertices, a Hilbert space for edges, and a graph gradient operator. The graph Laplacian operator can then be defined in terms of the two Hilbert spaces and the graph gradient operator. Each Hilbert space is defined by a different parameterized positive real-valued function, and the gradient operator is defined by a third parameterized positive real-valued function. The parameterizations provide the flexibility to learn the geometry of the vertex space, the geometry of the edge space, as well as the graph gradient operator from data.

The Dirichlet energy, which is the squared norm of graph gradient flows on Hilbert space of edges, provides a measure of smoothness on the graph. A natural formulation of semi-supervised learning (SSL) would be to minimize the Dirichlet energy to ensure smoothness, subject to label constraints at some of the vertices [33, 34, 35, 36]. We show that this may suffer from what we term as *over-smoothing during training (OST)*: changing the node features will not affect the output representation at the nodes during training if the Hilbert spaces and the gradient operator are not defined as functions of the node features. The independence of output from features can be mitigated in SSL by making the vertex and edge spaces as well as the gradient operator depend on the node features. However, we show that over-smoothing may still be present if we use graph representations obtained by minimizing the Dirichlet energy for tasks such as graph classification or graph regression. These problems do not have label constraints during the inference process and may converge to a constant function over all nodes regardless of the input features, a problem we call *over-smoothing during inference (OSI)*.

To overcome both over-smoothing issues, we derive our implicit graph diffusion layer through a constrained Dirichlet energy minimization problem which constrains the representation to be close to the node features. This provides a smooth representation, as defined by the learned edge geometry and gradient operator, while trading off the information in the node features, as measured by the learned vertex distance function. Furthermore, we demonstrate that with an appropriate hyperparameter that is larger than the largest eigenvalue of the parameterized graph Laplacian, our framework has a unique equilibrium and simply iterating the fixed-point equation will converge quickly to the equilibrium.

We validate the effectiveness of our models on a variety of benchmark datasets for node classification and graph classification tasks. The results show that our models have better performance than the leading explicit and implicit GNN baselines for both node classification and graph classification in most cases. Due to limited space, we defer a detailed discussion on related work and all proofs to the Appendix. Source code available at `https://github.com/guoji-fu/DIGNN`.

2 Preliminaries and Background

We use $\mathbb{R}_+ = \{x \in \mathbb{R} \mid x \geq 0\}$ to denote the space of non-negative real values and $\mathbb{R}_+^* = \{x \in \mathbb{R} \mid x > 0\}$ to denote the space of positive real values. We denote scalars by lower- or upper-case letters and vectors and matrices by lower- and upper-case boldface letters, respectively. Given a matrix $\mathbf{A}$, $\mathbf{A}_{i,:}$ is its i-th row vector and $\mathbf{A}_{:,j}$ is its j-column vector, we denote its transpose by $\mathbf{A}^\top$ and its Hadamard product with another matrix $\mathbf{B}$ by $\mathbf{A} \odot \mathbf{B}$, i.e., the element-wise multiplication of $\mathbf{A}$ and $\mathbf{B}$. We use $\|\mathbf{A}\|$ to denote the spectral norm of a matrix $\mathbf{A}$. A set $[1, 2, \ldots, N]$ is denoted as $[N]$. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ be an undirected graph where $\mathcal{V} = [N]$ denotes the set of vertices, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges, $\mathbf{A} \in \mathbb{R}_+^{N \times N}$ is the adjacency matrix and $A_{i,j} = A_{j,i}, A_{i,j} > 0$ for $[i, j] \in \mathcal{E}$, $A_{i,j} = 0$ otherwise. $\mathbf{D} \in \mathbb{R}_+^{N \times N} = \mathrm{diag}(D_1, \ldots, D_N)$ denotes the diagonal degree matrix with $D_i = \sum_j A_{i,j}, i = 1, \ldots, N$. The meaning of other notations can be inferred from the context.

2.1 Hilbert space of functions on the vertices and edges

Definition 1 (Inner Product on the Vertex Space)**.** *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and any functions $f : \mathcal{V} \mapsto \mathbb{R}, g : \mathcal{V} \mapsto \mathbb{R}$, the inner product on the function space $\mathbb{R}^{\mathcal{V}}$ on vertices $\mathcal{V}$ is defined as*

$$\langle f, g \rangle_{\mathcal{V}} = \sum_{i=1}^{N} f(i)g(i)\chi(i), \tag{1}$$

where $\chi : \mathcal{V} \mapsto \mathbb{R}_+$ is a positive real-valued function and is strictly positive on $\mathbb{R}_+^*$.

Definition 2 (Inner Product on the Edge Space)**.** *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and any functions $F : \mathcal{E} \mapsto \mathbb{R}, G : \mathcal{E} \mapsto \mathbb{R}$, the inner product on the function space $\mathbb{R}^{\mathcal{E}}$ on edges $\mathcal{E}$ is defined as*

$$\langle F, G \rangle_{\mathcal{E}} = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} F([i,j])G([i,j])\phi([i,j]), \tag{2}$$

where $\phi : \mathcal{E} \mapsto \mathbb{R}_+$ is a positive real-valued function and is strictly positive on $\mathbb{R}_+^*$.

The inner products on the vertex space and the edge space induce two norms $\| \cdot \|_{\mathcal{V}} = \sqrt{\langle \cdot, \cdot \rangle_{\mathcal{V}}}$ and $\| \cdot \|_{\mathcal{E}} = \sqrt{\langle \cdot, \cdot \rangle_{\mathcal{E}}}$, respectively. We denote by $\mathcal{H}(\mathcal{V}, \chi) = (\mathbb{R}^{\mathcal{V}}, \langle \cdot, \cdot \rangle_{\mathcal{V}})$ and $\mathcal{H}(\mathcal{E}, \phi) = (\mathbb{R}^{\mathcal{E}}, \langle \cdot, \cdot \rangle_{\mathcal{E}})$ the corresponding Hilbert spaces.

Remark 1 (The cases for vector functions)**.** *If f and g are vector functions, i.e., $f, g : \mathcal{V} \mapsto \mathbb{R}^d$ for some $d \in \mathbb{N}$, then the inner product on the function space on vertices $\mathcal{V}$ is given by $\langle f, g \rangle_{\mathcal{V}} = \sum_{i=1}^{N} \langle f(i), g(i) \rangle \chi(i)$. Similarly, the inner product on the function space on edges $\mathcal{E}$ for vector functions $F, G : \mathcal{E} \mapsto \mathbb{R}^c$ is given by $\langle F, G \rangle_{\mathcal{E}} = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \langle F([i,j]), G([i,j]) \rangle \phi([i,j])$.*

2.2 Parameterized graph gradient and graph divergence

Definition 3 (Graph Gradient)**.** *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a function $f : \mathcal{V} \mapsto \mathbb{R}$, the graph gradient $\nabla : \mathcal{H}(\mathcal{V}, \chi) \mapsto \mathcal{H}(\mathcal{E}, \phi)$ is defined as follows:*

$$(\nabla f)([i,j]) = \varphi([i,j])(f(j) - f(i)), \text{ for all } [i,j] \in \mathcal{E}, \tag{3}$$

where $\varphi : \mathcal{E} \mapsto \mathbb{R}_+^*$ is a strictly positive real-valued function, i.e., it is strictly positive on $\mathbb{R}_+^*$.

Intuitively, the graph gradient $(\nabla f)([i,j])$ measures the variation of f on an edge $[i,j] \in \mathcal{E}$. We define the graph divergence as the adjoint of the graph gradient:

Definition 4 (Graph Divergence)**.** *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and any functions $f \in \mathcal{H}(\mathcal{V}, \chi) : \mathcal{V} \mapsto \mathbb{R}, g \in \mathcal{H}(\mathcal{E}, \phi) : \mathbb{E} \mapsto \mathbb{R}$, the graph divergence $\mathrm{div} : \mathcal{H}(\mathcal{E}, \phi) \mapsto \mathcal{H}(\mathcal{V}, \chi)$ is defined by*

$$\langle \nabla f, g \rangle_{\mathcal{E}} = \langle f, -\mathrm{div} g \rangle_{\mathcal{V}}. \tag{4}$$

Remark 2. *Note that f in Definition 3 and g in Definition 4 can be some vector functions $f : \mathcal{V} \mapsto \mathbb{R}^d, g : \mathcal{E} \mapsto \mathbb{R}^d$ for some $d \in \mathbb{N}$. Extending the definitions of the difference operator and its adjoint and the graph Laplacian to vector functions should be straightforward by using the inner products on vertex space and edge space for vector functions.*

Lemma 1. *Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a function $g : \mathcal{E} \mapsto \mathbb{R}$, the graph divergence $\mathrm{div} : \mathcal{H}(\mathcal{E}, \phi) \mapsto \mathcal{H}(\mathcal{V}, \chi)$ is explicitly given by*

$$(\mathrm{div} g)(i) = \frac{1}{2\chi(i)} \sum_{j=1}^{N} \varphi([i,j])\phi([i,j]) \left(g([i,j]) - g([j,i]) \right). \tag{5}$$

For the proof, see Appendix B.1. Intuitively, $(\mathrm{div} g)(i)$ measures the inflows of g towards vertex i and the outflows from i.

2.3 Parameterized graph Laplacian operator

Definition 5 (Graph Laplacian Operator)**.** *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, positive real-valued functions $\chi : \mathcal{V} \mapsto \mathbb{R}_+, \phi : \mathcal{E} \mapsto \mathbb{R}_+, \varphi : \mathcal{E} \mapsto \mathbb{R}_+^*$, graph gradient ∇ and graph divergence operator div, the graph Laplacian operator $\Delta : \mathcal{H}(\mathcal{V}, \chi) \mapsto \mathcal{H}(\mathcal{V}, \chi)$ is defined as*

$$\Delta = -\mathrm{div} \nabla. \tag{6}$$

Table 1: Graph Laplacian Operators.

Type	Hilbert Spaces	Positive real-valued function
Unnormalized $\Delta^{(\mathrm{un})}$	Fixed	$\chi(i) = 1, \varphi^2([i,j])\phi([i,j]) = A_{i,j}$
Random walk $\Delta^{(\mathrm{rw})}$	Fixed	$\chi(i) = D_i, \varphi^2([i,j])\phi([i,j]) = A_{i,j}$
Normalized[1] $\Delta^{(\mathrm{n})}$	Fixed	$\chi(i) = \sqrt{D_i}, \phi^2([i,j])\phi([i,j]) = A_{i,j}$
Parameterized Δ_Φ	Learnable	$\chi(i) = D_i \tanh(\lVert\Theta_\chi \mathbf{x}_i\rVert)$ $\phi([i,j]) = \tanh(\lvert(\Theta_\phi\Theta_\chi\mathbf{x}_i)^\top\Theta_\phi\Theta_\chi\mathbf{x}_j\rvert)$ $\varphi([i,j]) = \sqrt{A_{i,j}\tanh((\lVert\Theta_\varphi(\mathbf{x}_i-\mathbf{x}_j)\rVert+\epsilon)^{-1})}$

In the following, Lemma 2 (For the proof, see Appendix B.2) gives the explicit expression of Δ:

Lemma 2. *Given a function $f : \mathcal{V} \mapsto \mathbb{R}$, the graph Laplacian $\Delta : \mathcal{H}(\mathcal{V},\chi) \mapsto \mathcal{H}(\mathcal{V},\chi)$ is given by:*

$$(\Delta f)(i) = \frac{1}{\chi(i)} \sum_{j=1}^{N} \varphi([i,j])^2 \phi([i,j]) \left(f(i) - f(j)\right). \tag{7}$$

We also demonstrate that for any positive real-valued functions χ, ϕ, φ, the graph Laplacian is self-adjoint and positive semi-definite by Proposition 8 in Appendix B.8.

Remark 3 (Compare to canonical Laplacians)**.** *Canonical graph Laplacian operators, such as unnormalized Laplacian, random walk Laplacian, and normalized Laplacian are special cases in our definition, whose corresponding Hilbert spaces on vertices and edges and graph gradient operator are fixed as summarized in Table 1. We provide a detailed discussion on these operators in Appendix D.1.*

2.4 Graph neural Laplacian

Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ with node embeddings $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)^\top \in \mathbb{R}^{N\times d}$ with each row $\mathbf{x}_i \in \mathbb{R}^d$ a feature vector, we design the three positive real-valued functions χ, ϕ, φ as

$$\chi(i) = D_i \tanh(\lVert\Theta_\chi \mathbf{x}_i\rVert), \quad \text{for all } i \in \mathcal{V}, \tag{8}$$

$$\phi([i,j]) = \tanh(\lvert(\Theta_\phi\Theta_\chi\mathbf{x}_i)^\top(\Theta_\phi\Theta_\chi\mathbf{x}_j)\rvert), \quad \text{for all } [i,j] \in \mathcal{E} \tag{9}$$

$$\varphi([i,j]) = \sqrt{A_{i,j}\tanh(\lVert\Theta_\varphi((\mathbf{x}_i-\mathbf{x}_j)\rVert+\epsilon)^{-1})}, \quad \text{for all } [i,j] \in \mathcal{E} \tag{10}$$

where $\epsilon > 0$ is a small positive value used to avoid dividing zero for neighbor nodes with identical embeddings. $\Theta_\chi, \Theta_\phi, \Theta_\varphi$ are matrices of learnable parameters. Note that $\phi([i,j]) = \phi([j,i])$ and $\varphi([i,j]) = \varphi([j,i])$ for all $[i,j] \in \mathcal{E}$. Then the graph neural Laplacian $\Delta_\Phi : \mathcal{H}(\mathcal{V},\chi) \mapsto \mathcal{H}(\mathcal{V},\chi)$ is given by: for all $i \in \mathcal{V}$,

$$(\Delta_\Phi f)(i) = \sum_{j=1}^{N} \frac{A_{i,j}}{D_i} \frac{\tanh(\lVert\Theta_\varphi((\mathbf{x}_i-\mathbf{x}_j)\rVert+\epsilon)^{-1})\tanh(\lvert(\Theta_\phi\Theta_\chi\mathbf{x}_i)^\top(\Theta_\phi\Theta_\chi\mathbf{x}_j)\rvert)}{\tanh(\lVert\Theta_\chi\mathbf{x}_i\rVert)}(f(i)-f(j)). \tag{11}$$

It is easy to verify that Equations (8) to (10) satisfy the requirements that χ, ϕ, φ are strictly positive on $\mathbb{R}_+^*$. We adopt $\tanh(\cdot)$ as the nonlinearity as it satisfies the requirements and can be upper-bounded which will be helpful for spectral analysis and in ensuring convergence of the induced implicit layer. χ (Eq. (8)) and ϕ (Eq. (9)) are used to learn the Hilbert spaces on vertices and edges, respectively. φ (Eq. (10)) is used to learn the diffusivity of the graph gradient operator. Intuitively, the forms of Equations (8) to (10) have the following properties: influence of node i on the vertex space geometry would be reduced if $\lVert\Theta_\chi\mathbf{x}_i\rVert$ is small by Eq. (8); if $\lVert\Theta_\chi\mathbf{x}_i\rVert$ is small, edges to node i would have small weights but if $\lVert\Theta_\chi\mathbf{x}_i\rVert$ is large, the weight would depend on the learned similarities of the two nodes by Eq. (9); Eq. (10) focuses on the difference between nodes and would impose larger diffusion weights for similar neighborhood nodes and smaller weights for neighbors that are dissimilar. Notably, here we just provide three feasible choices with additional expected properties for χ, ϕ, φ. In general, other positive real-valued functions can be designed in the graph neural Laplacian to exploit problem-specific properties.

[1]The graph gradient for the normalized case is defined as $(\nabla f)([i,j]) = \varphi([i,j])(f(j)/\sqrt{D_j} - f(i)/\sqrt{D_i})$.

Theorem 3 (Spectral Range of Δ_Φ)**.** *Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ with node embeddings $\mathbf{X} \in \mathbb{R}^{N\times d}$, let positive real-valued functions χ, ϕ, φ of the graph neural Laplacian Δ_Φ be parameterized by Equations (8) to (10), respectively. Suppose that $\|\mathbf{x}_i\| \le c_X$ for all $i \in \mathcal{V}$ and the parameters Θ_χ, Θ_ϕ can be bounded by $\|\Theta_\chi\| \le c_\chi, \|\Theta_\phi\| \le c_\phi$ for some $c_X, c_\chi, c_\phi \in \mathbb{R}^*_+$. If $\mathcal{G}$ is connected and λ is a eigenvalue associated with the eigenvector $\mathbf{u}$ of Δ_Φ, then*

$$0 \le \lambda \le 2c_\phi^2 c_\chi c_X \cosh(c_\chi c_X).$$

For the proof, see Appendix B.3. The size of the largest eigenvalue is essential for determining the convergence of the implicit GNN. Theorem 3 shows that the eigenvalues will be small if c_X, $\|\Theta_\chi\|$ and $\|\Theta_\phi\|$ are kept small.

3 An Optimization View for Implicit Graph Neural Diffusions and Over-Smoothing Issues

3.1 Parameterized Dirichlet energy

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define the parameterized Dirichlet energy as the square norm of graph gradient flows on the Hilbert space of edges (See Appendix C.1 for detailed derivation):

$$\mathcal{S}(f) = \|\nabla f\|^2_{\mathcal{E}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \varphi([i,j])^2\phi([i,j])\|f(j) - f(i)\|^2. \tag{12}$$

$\mathcal{S}(f)$ measures the variation of f over the whole graph $\mathcal{G}$ and lower energy graphs are smoother. We have $\left.\frac{\mathrm{d}\mathcal{S}(f)}{\mathrm{d}f}\right|_i = 2\chi(i)(\Delta f)(i)$ by Lemma 9 as given in Appendix B.9.

3.2 Energy minimization and over-smoothing

Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ with node features $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top \in \mathbb{R}^{N\times d}$, assume that we have a subset of constrained nodes $\mathcal{I} \subseteq \mathcal{V}$ where each node $i \in \mathcal{I}$ is provided with a target vector $\mathbf{y}(i) \in \mathbb{R}^c$. We aim to learn a function $f = (f(1), \ldots, f(N))^\top \in \mathbb{R}^{N\times c}$ to predict the value of all the nodes. To this end, a natural formulation would be to minimize the Dirichlet energy to ensure smoothness, subject to the function being close to the targets at the constrained nodes:

$$\begin{aligned}\inf_f \mathcal{L}(f) &= \inf_f \left[\mathcal{S}(f) + \mu\sum_{i\in\mathcal{I}}\|f(i) - \mathbf{y}(i)\|^2_{\mathcal{V}}\right] \\ &= \inf_f \left[\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\varphi([i,j])^2\phi([i,j])\|f(j) - f(i)\|^2 + \mu\sum_{i\in\mathcal{I}}\|f(i) - \mathbf{y}(i)\|^2_{\mathcal{V}}\right].\end{aligned} \tag{13}$$

Denote $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_N) \in \mathbb{R}^N$ and $\mathbf{Y}' = (\mathbf{y}'(1), \ldots, \mathbf{y}'(N))^\top \in \mathbb{R}^{N\times c}$. If $i \in \mathcal{I}$, let $\delta_i = 1, \mathbf{y}'(i) = \mathbf{y}(i)$ and $\delta_i = 0, \mathbf{y}'(i) = \mathbf{0}$ otherwise. Let $\widehat{\mathbf{D}} = \mathrm{diag}(\widehat{D}_1, \ldots, \widehat{D}_N) \in \mathbb{R}^{N\times N}$ with $\widehat{D}_i = \sum_k \varphi(i,k)^2\phi([i,k])$, $\mathbf{P} \in \mathbb{R}^{N\times N}$ with $P_{i,j} = \varphi([i,j])^2\phi([i,j])/\widehat{D}_i$, $\mathbf{C} \in \mathbb{R}^{N\times N}$ with $\mathbf{C}_{i,:} = (1-\delta_i)\mathbf{P}_{i,:} - \delta_i\frac{1}{\mu}\Delta_{i,:}$.

The solution of Eq. (13) satisfies (see Appendix C.2 for detailed derivation):

$$f = \mathbf{Y}' + \mathbf{C}f. \tag{14}$$

Lemma 4 (Convergence and Uniqueness Analysis)**.** *Let γ_{max} be the largest singular value of $\mathbf{C}$. The fixed-point equilibrium equation $f = \mathbf{Y}' + \mathbf{C}f$ has a unique solution if $\gamma_{max} < 1$. The solution can be obtained by iterating the equation:*

$$f^{(t+1)} = \mathbf{Y}' + \mathbf{C}f^{(t)}, \quad t = 0, 1, \ldots. \tag{15}$$

to obtain $f^\star = \lim_{t\to\infty} f^{(t)}$ which is independent of the initial state $f^{(0)}$.

For the proof, see Appendix B.4. In the following, we describe the conditions that lead to two types of over-smoothing phenomena in learning problems, namely over-smoothing during training (OST) and over-smoothing during inference (OSI), as outlined in Corollary 5 and Lemma 6 respectively.

Corollary 5 (Over-Smoothing during Training (OST) Condition)**.** *Assume that the initial state $f^{(0)}$ at node i is initialized as a function of $\mathbf{x}_i$. Assume further that χ, ϕ, φ are not functions of $\mathbf{x}_i$, e.g., Δ is the unnormalized Laplacian or random walk Laplacian. Let γ_{max} be the largest singular value of $\mathbf{C}$. Then, if $\gamma_{max} < 1$, the equilibrium of the fixed point equation Eq. (14) is independent of the node features $\mathbf{x}_i$, i.e. the same equilibrium solution is obtained even if the node features $\mathbf{x}_i$ are changed.*

For the proof, see Appendix B.5. We describe how OST can be problematic for semi-supervised learning where some of the nodes are labeled and we would like to predict the labels of the remaining nodes. In the simplest case, we use $f(i)$ to directly predict the target $y(i)$ and do the semi-supervised learning by solving Eq. (13). Corollary 5 indicates that changing the node features, e.g. through feature learning, will not affect the output representation at the nodes if the vertex and edge Hilbert spaces as well as the gradient operator are not defined as functions of the node features. This implies that the node features are not used at all in learning the prediction function, which would be undesirable if the node features contain useful information for the problem. The independence of output from features can be mitigated in semi-supervised learning by making the vertex and edge spaces as well as the gradient operator depend on the node features. However, the dependence is indirect, through the edge and gradient, making the strength of dependence on the features more difficult to control. Furthermore, over-smoothing may still be present if we use graph representations obtained by minimizing the Dirichlet energy for tasks such as graph classification or graph regression which do not have any constrained nodes during the inference process. As shown by Lemma 6 (see Appendix B.6 for the proof), when there are no constrained nodes in Eq. (13), the equilibrium of Eq. (14) is a constant function over all nodes, which drastically reduces what it is able to represent.

Lemma 6 (Over-Smoothing during Inference (OSI) Condition)**.** *When $\mathcal{I} = \emptyset$, i.e., there is no constrained nodes in Eq. (13), we have $\mathbf{C} = \mathbf{P}$ and $\mathbf{y}'(i) = \mathbf{0}, \forall i \in \mathcal{V}$. $\mathbf{P}$ is a Markov matrix and the solution of Eq. (13) satisfies $f = \mathbf{P}f$. If the graph $\mathcal{G}$ is connected and non-bipartite, i,e., the Markov chain corresponding to $\mathbf{P}$ is irreducible and aperiodic, the equilibrium $f^\star(i)$ for all nodes are identical and not unique, i.e., $f^\star(i) = \mathbf{v}, \forall i \in \mathcal{V}$, where $\mathbf{v}$ is any vector in $\mathbb{R}^c$ and is independent of the labels. Iterating $f^{(t+1)} = \mathbf{P}f^{(t)}, t = 0, 1, \ldots,$ with initial state $f^{(0)} \in \mathbb{R}^{N\times c}$ will converge to*

$$f^\star(i) = (\lim_{t\to\infty} \mathbf{P}^t f^{(0)})(i) = (\boldsymbol{\pi} f^{(0)})^\top, \quad \textit{for all } i \in \mathcal{V}, \tag{16}$$

where $\boldsymbol{\pi} = (\widehat{D}_1/\sum_j \widehat{D}_j, \ldots, \widehat{D}_N/\sum_j \widehat{D}_j) \in \mathbb{R}^{1\times N}$ is the stationary distribution of $\mathbf{P}$.

Remark 4 (Compare Lemma 6 to the result in [29])**.** *A similar result on the OSI phenomenon in the GRAND architecture [26] was previously provided in [29] (see Proposition 3 in [29], which presents a random walk interpretation of GRAND). We refer to Appendix A for a more detailed comparison between our work and [29].*

3.3 Dirichlet implicit graph neural networks

We would like the output of the implicit GNN to depend on the features $\mathbf{X}$; when it does not, we view the problem as over-smoothing. A straightforward way to guarantee that the output of the implicit GNN depends on $\mathbf{X}$ is to ensure that each node is constrained and the target constraint $\mathbf{y}(i)$ is a function of $\mathbf{x}_i$. Further, by adjusting the hyperparameter μ, we can control the strength of the constraints, allowing a tradeoff between the exploitation of node features and smoothing. We incorporate this idea in the design of our Dirichlet Implicit Graph Neural Network, termed DIGNN.

The architecture of DIGNN is given as follows:

$$\widetilde{\mathbf{X}} = h_{\Theta^{(1)}}(\mathbf{A}, \mathbf{X}), \tag{17}$$

$$\mathbf{Z} = \widetilde{\mathbf{X}} - \frac{1}{\mu}\Delta\mathbf{Z}, \tag{18}$$

$$\widehat{\mathbf{Y}} = h_{\Theta^{(2)}}(\mathbf{Z}). \tag{19}$$

The first layer Eq. (17) serves as a feature preprocessing unit, which could be merely a multi-layer perceptron that operates on the node features and the adjacency matrix. The second layer Eq. (18) is the implicit graph diffusion layer induced by the fixed-point equilibrium equation Eq. (14) with source constraints on the vertex features. The last layer Eq. (19) is the output layer. The subsequent corollary provides the tractable well-posedness conditions of our implicit layer and outlines an iterative algorithm designed to obtain the equilibrium, complete with an assurance of convergence.

Corollary 7 (Tractable Well-Posedness Condition and Convergence Rate)**.** *Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ with node embeddings $\widetilde{\mathbf{X}} = (\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_N)^\top \in \mathbb{R}^{N\times d}$, Let λ_{max} be the largest eigenvalues of the matrix Δ. Suppose that $\mathcal{G}$ is connected, the fixed-point equilibrium equation $\mathbf{Z} = \widetilde{\mathbf{X}} - \frac{1}{\mu}\Delta\mathbf{Z}$ has a unique solution if $\mu > \lambda_{max}$. The solution can be obtained by iterating:*

$$\mathbf{Z}^{(t+1)} = \widetilde{\mathbf{X}} - \frac{1}{\mu}\Delta\mathbf{Z}^{(t)}, \quad \textit{with } \mathbf{Z}^{(0)} = \mathbf{0}, \quad t = 0, 1, \ldots.$$

Therefore, $\mathbf{Z}^\star = \lim_{t\to\infty}\mathbf{Z}^{(t)}$. Suppose that $\|\mathbf{Z}^\star\| \le c_Z \in \mathbb{R}_+^*$, then $\forall t \ge 1$, $\|\mathbf{Z}^{(t)} - \mathbf{Z}^\star\| \le c_Z\left(\lambda_{max}/\mu\right)^t$. Specifically,

1. *If Δ is the random walk Laplacian, i.e., $\Delta = \Delta^{(rw)}$, $\mu > 2$ and $\|\mathbf{Z}^{(t)} - \mathbf{Z}^\star\| \le c_Z\left(2/\mu\right)^t$;*
2. *If Δ is learnable and $\Delta = \Delta_\Phi$ as given in Eq. (11), suppose that $\|\widetilde{\mathbf{x}}_i\| \le c_X$ for all $i \in \mathcal{V}$, the parameters Θ_χ, Θ_ϕ can be bounded by $\|\Theta_\chi\| \le c_\chi$, $\|\Theta_\phi\| \le c_\phi$ and $c_X, c_\chi, c_\phi \in \mathbb{R}_+^*$, we have $\mu > 2c_\phi^2 c_\chi c_X \cosh(c_\chi c_X)$ and $\|\mathbf{Z}^{(t)} - \mathbf{Z}^\star\| \le c_Z(2c_\phi^2 c_\chi c_X \cosh(c_\chi c_X)/\mu)^t$.*

For the proof, see Appendix B.7. Corollary 7 shows that the selection of the hyperparameter μ is linked to the upper bound β of the norm of the node embeddings when we apply $\Delta = \Delta_\Phi$. Hence, to maintain a minimal β, we can perform batch normalization on $\widetilde{\mathbf{X}}$ between the first and second layers.

3.4 Training of DIGNN

With appropriate hyperparameter $\lambda_{\max} < \mu$, Corollary 7 illustrates that the equilibrium of the fixed-point equation Eq. (18) can be obtained by iterating the Corollary 7. Therefore, for the forward evaluation, we simply iterate Corollary 7. For the backward pass, we can use implicit differentiation [37, 20] to compute the gradients of trainable parameters by directly differentiating through the equilibrium. See Appendix C.3 for details of derivation for backward gradient computation.

4 Empirical Studies

In this section, we conduct comprehensive experiments to evaluate the effectiveness of our models against both implicit and explicit GNNs on benchmark datasets for node classification and graph classification. We refer to Appendix E for the details of data statistics and experimental setup. In the experiments, we adopt the random walk Laplacian $\Delta^{(\mathrm{rw})}$ and the graph neural Laplacian Δ_Φ for DIGNN, which are denoted by DIGNN-$\Delta^{(\mathrm{rw})}$ (discussed in Appendix D.1) and DIGNN-Δ_Φ (Eq. (11)), respectively. Our models are implemented based on the PyTorch Geometric library [38].

4.1 Semi-supervised node classification

Datasets. We conduct node classification experiments on five relatively large heterophilic datasets: Chameleon, Squirrel [39], Penn94, Cornell5, and Amherst41 [40], and three commonly used homophilic datasets: Cora, CiteSeer, and PubMed [41]. We also evaluate our model on PPI dataset [42] for multi-label multi-graph inductive learning. For Chameleon, Squirrel, Cora, CiteSeer, and PubMed, we use standard train/validation/test splits as in [43]. For Penn94, Cornell15, and Amherst41, we follow the same data splits as in [40]. For PPI, we use the standard data splits as in [3].

Baselines. We compare our models with several representative explicit GNNs such as GCN [2], GAT [5], GCNII [44], H2GCN [45], APPNP [8], LINKX [40], and implicit GNNs, including IGNN [22], GRAND-l [26], MGNNI [25], GIND [24]. The results of all baselines except for GRAND-l on Chameleon, Squirrel, Cora, CiteSeer, PubMed, and PPI are borrowed from [25, 24, 40], while the results for GRAND-l were obtained by us using their officially published source code. For Penn94, Cornell5, and Amherst41, all baselines were run by us to obtain their results.

Results. The results in Table 2 demonstrate that DIGNN-Δ_Φ substantially surpasses both explicit and implicit baselines on all heterophilic datasets. In particular, the performance on Chameleon and Squirrel datasets is noteworthy, with our model DIGNN-Δ_Φ considerably improving the state-of-the-art (SOTA) accuracy of LINKX on Chameleon from 68.42% to 79.89% and on Squirrel from 61.81% to 74.96%. In contrast, the SOTA implicit model GIND only managed to achieve 66.82% and 56.71%. The outcomes from Chameleon and Squirrel also show that APPNP and GCNII, which incorporate T-hop neighbors to mitigate over-smoothing, perform better than GCN and GAT but are substantially outperformed by most implicit GNNs. These findings emphasize the importance of long-range dependencies for learning on the Chameleon and Squirrel datasets and validate the clear advantages of our models over all baselines in effectively leveraging long-range dependencies.

Table 2: Results on heterophilic graph datasets: mean accuracy (%) (standard deviation (stdev)) over 10 random data splits. Best results outlined in bold. OOT denotes out of time.

	Chameleon	Squirrel	Penn94	Cornell5	Amherst41
#Nodes	2,227	5,201	41,554	18,660	2,235
#Edges	31,421	198,493	1,362,229	790,777	90,954
#Clases	5	5	2	2	2
GCN	42.34 (2.77)	29.00 (1.10)	82.47 (0.27)	80.15 (0.37)	81.41 (1.70)
GAT	46.03 (2.51)	30.51 (1.28)	81.53 (0.55)	78.96 (1.57)	79.33 (2.09)
GCNII	48.59 (1.88)	32.20 (1.06)	82.92 (0.59)	78.85 (0.78)	76.02 (1.38)
H2GCN	60.30 (2.31)	40.75 (1.44)	81.31 (0.60)	78.46 (0.75)	79.64 (1.63)
APPNP	43.85 (2.43)	30.67 (1.06)	74.79 (0.43)	73.23 (1.08)	68.34 (2.92)
LINKX	68.42 (1.38)	61.81 (1.80)	84.71 (0.52)	83.46 (0.61)	81.73 (1.94)
IGNN	41.38 (2.53)	24.99 (2.11)	-	-	-
GRAND-l*	42.39 (4.61)	34.57 (0.89)	81.77 (0.39)	81.22 (0.81)	80.07 (1.99)
MGNNI	63.93 (2.21)	54.50 (2.10)	OOT	78.11 (0.67)	75.59 (1.65)
GIND	66.82 (2.37)	56.71 (2.07)	76.29 (0.78)	73.91 (0.92)	72.24 (2.20)
DIGNN-$\Delta^{(\mathrm{rw})}$	79.21 (1.22)	73.76 (2.19)	83.17 (0.33)	82.27 (0.56)	80.78 (2.17)
DIGNN-Δ_Φ	**79.89** (1.51)	**74.96** (1.77)	**86.19** (0.18)	**84.44** (0.59)	**83.34** (1.21)

Table 3: Results on homophilic node classification datasets: mean accuracy (%). Best results outlined in bold.

	Cora	CiteSeer	PubMed
GCN	85.77	73.68	88.13
GAT	86.37	74.32	87.62
APPNP	87.87	76.53	89.40
Geom-GCN	85.27	**77.99**	**90.05**
GCNII	**88.49**	77.08	89.57
H2GCN	87.87	77.11	89.49
IGNN	85.80	75.24	87.66
EIGNN	85.89	75.31	87.92
GRAND-l*	87.02	75.79	87.08
MGNNI	83.37	75.57	88.03
GIND	88.25	76.81	89.22
DIGNN-$\Delta^{(\mathrm{rw})}$	85.01	75.22	88.65
DIGNN-Δ_Φ	86.68	76.98	88.60

Table 4: Results of multi-label node classification on PPI. Best results outlined in bold.

Method	Micro-F1 (%)
GCN	59.2
GraphSAGE	78.6
GAT	97.3
JKNet	97.6
GCNII	**99.5**
IGNN	97.0
EIGNN	98.0
MGNNI	98.7
GIND	98.4
DIGNN-$\Delta^{(\mathrm{rw})}$	98.9
DIGNN-Δ_Φ	99.1

On the other hand, the results for Penn94, Cornell5, and Amherst41 illustrate that DIGNN-Δ_Φ consistently surpasses all explicit and implicit baselines. However, implicit baselines, APPNP, and GCNII, which are either able to consider a larger range of neighbors or to mitigate over-smoothing, do not outperform GCN, GAT, and LINKX. It indicates that for these three heterophilic datasets, long-range dependencies may not be as beneficial as they are for Chameleon and Squirrel. Instead, the geometry of vertex and edge spaces may play a more significant role. The conclusion can be reinforced by the observation that DIGNN-Δ_Φ surpasses DIGNN-$\Delta^{(\mathrm{rw})}$, which adopts fixed Hilbert spaces of functions on the vertices and edges.

Additionally, Table 3 shows that our models obtain comparable accuracy to all baselines on homophilic datasets Cora, CiteSeer, and PubMed where long-range dependencies might not significantly enhance the prediction performance. The results presented in Table 4 for PPI show that DIGNNs ($\Delta^{(\mathrm{rw})}$ and Δ_Φ) achieve Micro-F1 scores of 98.9% and 99.1% respectively. Our models surpass most explicit and implicit baselines and are close to the SOTA GCNII. It suggests that DIGNNs apply to multi-label multi-graph inductive learning.

4.2 Graph classification

Datasets. For graph classification tasks, we use four bioinformatics datasets [46], MUTAG, PTC, PROTEINS, NCI1, and two social network datasets [46], IMDB-BINARY and IMDB-MULTI. We use the standard train/validation/test splits as in [47].

Table 5: Results on graph datasets: accuracy (%) (stdev) over 10 folds. Best results outlined in bold.

	MUTAG	PTC	PROTEINS	NCI1	IMDB-B	IMDB-M
#Graphs	188	344	1113	4110	1000	1500
Avg #Nodes	17.9	25.5	39.1	29.8	19.8	13.0
#Classes	2	2	2	2	2	3
GCN	85.6 (5.8)	64.2 (4.3)	76.0 (3.2)	80.2 (2.0)	-	-
GIN	89.0 (6.0)	63.7 (8.2)	75.9 (3.8)	**82.7** (1.6)	75.1 (5.1)	52.3 (2.8)
DGCNN	85.8	58.6	75.5	74.4	70.0	47.8
FDGNN	88.5 (3.8)	63.4 (5.4)	76.8 (2.9)	77.8 (1.6)	72.4 (3.6)	50.0 (1.3)
IGNN	89.3 (6.7)	70.1 (5.6)	77.7 (3.4)	80.5 (1.9)	-	-
EIGNN	88.9 (1.1)	69.8 (5.3)	75.9 (6.4)	77.5 (2.2)	72.3 (4.3)	52.1 (2.9)
CGS	89.4 (5.6)	64.7 (6.4)	76.3 (6.3)	77.2 (2.0)	73.1 (3.3)	51.1 (2.2)
MGNNI	91.9 (5.5)	72.1 (2.8)	79.2 (2.9)	78.9 (2.1)	75.8 (3.4)	53.5 (2.8)
GIND	89.3 (7.4)	66.9 (6.6)	77.2 (2.9)	78.8 (1.7)	-	-
DIGNN-$\Delta^{(\mathrm{rw})}$	90.8 (7.4)	74.1 (5.3)	79.9 (2.9)	77.7 (1.4)	77.7 (3.2)	53.9 (3.7)
DIGNN-Δ_Φ	**94.6** (4.9)	**76.5** (3.7)	**81.3** (3.2)	77.9 (2.0	**78.4** (3.6)	**54.2** (2.9)

Baselines. We include the baselines that also have reported results on the chosen datasets, including explicit models such as GCN, GIN [47], DGCNN [48], FDGNN [49], and implicit models, such as IGNN, CGS [32], MGNNI, GIND. The results for all baselines are borrowed from [25] and [24].

Results. As shown in Table 5, DIGNN-Δ_Φ substantially outperforms all baselines on all graph classification datasets except for NCI1. For NCI1, DIGNNs ($\Delta^{(\mathrm{rw})}$ and Δ_Φ) are comparable to the baselines. MGNNI and GIND are better than explicit baselines on most datasets. It suggests that long-range dependencies could be helpful for graph classification tasks. Moreover, DIGNN-Δ_Φ outperforms $\Delta^{(\mathrm{rw})}$ and implicit baselines on all datasets. The results demonstrate that the underlying geometry of vertex and edge spaces is also important for graph classification when generalizing to unseen testing graphs. These observations again confirm the capability of DIGNN-Δ_Φ in effectively capturing long-range dependencies and the underlying graph geometry.

4.3 Over-smoothing analysis

We carry out node classification experiments on Cora, CiteSeer, and PubMed to demonstrate our model would not suffer from the over-smoothing issue and performs well with very deep implicit layers. Figure 1 shows that as the number of layers increases our model maintains good performance whilst GCN degrades catastrophically. Additionally, APPNP performs more stably than GCN because it can incorporate T-hop neighbors to mitigate over-smoothing. Nevertheless, its performance tends to decline when the architecture becomes exceedingly deep, as demonstrated by the outcomes from Cora and PubMed.

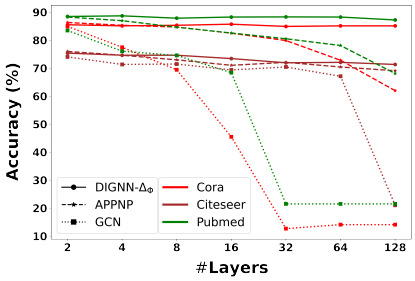


Figure 1: Performance of architectures with different depths. Best viewed in colors.

5 Conclusion

This paper aims to equip implicit GNNs with the ability to learn the graph metric and tackle over-smoothing issues. To this end, we propose a geometric framework for designing implicit graph neural diffusion layers based on the parameterized graph Laplacian operator. Our framework not only learns the geometry of vertex and edge spaces but also the graph gradient operator from the data. By viewing implicit GNN layers as the fixed-point solution of a Dirichlet energy minimization problem, we identify conditions that may lead to over-smoothing. To overcome the over-smoothing issue, we design our implicit graph diffusion layer as the solution to a Dirichlet energy minimization problem with constraints on vertex features, facilitating a balance between smoothing and preservation of node feature information. We demonstrate that with an appropriate hyperparameter set to be greater than the largest eigenvalue of the parameterized graph Laplacian, our framework ensures a unique equilibrium and quick convergence. Comparative studies on benchmark datasets for node and graph classification tasks show that our models outperform leading implicit and explicit GNNs, with notable accuracy enhancements observed in several datasets.

References

[1] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems 29, NIPS 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3837–3845, 2016.

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[3] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds.), pp. 1024–1034, 2017.

[4] F. Wu, A. H. S. Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 6861–6871, 2019.

[5] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[6] K. K. Thekumparampil, C. Wang, S. Oh, and L. Li, "Attention-based graph neural network for semi-supervised learning," *CoRR*, vol. abs/1803.03735, 2018.

[7] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272, PMLR, 2017.

[8] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[9] E. Chien, J. Peng, P. Li, and O. Milenkovic, "Adaptive universal generalized pagerank graph neural network," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021.

[10] G. Fu, P. Zhao, and Y. Bian, "p-laplacian based graph neural networks," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA* (K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, eds.), vol. 162 of *Proceedings of Machine Learning Research*, pp. 6878–6917, PMLR, 2022.

[11] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, "Deepinf: Social influence prediction with deep learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018* (Y. Guo and F. Farooq, eds.), pp. 2110–2119, ACM, 2018.

[12] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018* (Y. Guo and F. Farooq, eds.), pp. 974–983, ACM, 2018.

[13] W. Fan, Y. Ma, Q. Li, Y. He, Y. E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019* (L. Liu, R. W. White, A. Mantrach, F. Silvestri, J. J. McAuley, R. Baeza-Yates, and L. Zia, eds.), pp. 417–426, ACM, 2019.

[14] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017* (M. Palmer, R. Hwa, and S. Riedel, eds.), pp. 1506–1515, Association for Computational Linguistics, 2017.

[15] H. Gao, Y. Chen, and S. Ji, "Learning graph pooling and hybrid convolutional operations for text representations," in *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019* (L. Liu, R. W. White, A. Mantrach, F. Silvestri, J. J. McAuley, R. Baeza-Yates, and L. Zia, eds.), pp. 2743–2749, ACM, 2019.

[16] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in Neural Information Processing Systems 28, NeurIPS 2015 December 7-12, 2015, Montreal, Quebec, Canada* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 2224–2232, 2015.

[17] F. Wan, L. Hong, A. Xiao, T. Jiang, and J. Zeng, "Neodti: neural integration of neighbor information from a heterogeneous network for discovering new drug-target interactions," *Bioinform.*, vol. 35, no. 1, pp. 104–111, 2019.

[18] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI Conference on Artificial Intelligence, AAAI 2018, New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3538–3545, 2018.

[19] N. Keriven, "Not too little, not too much: a theoretical analysis of graph (over)smoothing," in *NeurIPS*, 2022.

[20] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds.), pp. 688–699, 2019.

[21] L. E. Ghaoui, F. Gu, B. Travacca, and A. Askari, "Implicit deep learning," *CoRR*, vol. abs/1908.06315, 2019.

[22] F. Gu, H. Chang, W. Zhu, S. Sojoudi, and L. E. Ghaoui, "Implicit graph neural networks," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[23] J. Liu, K. Kawaguchi, B. Hooi, Y. Wang, and X. Xiao, "EIGNN: efficient infinite-depth graph neural networks," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 18762–18773, 2021.

[24] Q. Chen, Y. Wang, Y. Wang, J. Yang, and Z. Lin, "Optimization-induced graph implicit nonlinear diffusion," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, vol. 162 of *Proceedings of Machine Learning Research*, pp. 3648–3661, PMLR, 2022.

[25] J. Liu, B. Hooi, K. Kawaguchi, and X. Xiao, "MGNNI: multiscale graph neural networks with implicit layers," in *NeurIPS*, 2022.

[26] B. Chamberlain, J. Rowbottom, M. I. Gorinova, M. M. Bronstein, S. Webb, and E. Rossi, "GRAND: graph neural diffusion," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol. 139 of *Proceedings of Machine Learning Research*, pp. 1407–1418, PMLR, 2021.

[27] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[28] Y. Wang, Y. Wang, J. Yang, and Z. Lin, "Dissecting the diffusion process in linear graph convolutional networks," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 5758–5769, 2021.

[29] M. Thorpe, T. M. Nguyen, H. Xia, T. Strohmer, A. L. Bertozzi, S. J. Osher, and B. Wang, "GRAND++: graph neural diffusion with A source term," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, OpenReview.net, 2022.

[30] C. Bodnar, F. D. Giovanni, B. P. Chamberlain, P. Liò, and M. M. Bronstein, "Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns," *CoRR*, vol. abs/2202.04579, 2022.

[31] L. A. C. Xhonneux, M. Qu, and J. Tang, "Continuous graph neural networks," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 10432–10441, PMLR, 2020.

[32] J. Park, J. Choo, and J. Park, "Convergent graph solvers," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, OpenReview.net, 2022.

[33] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems 16, NIPS 2004, December 8-13, 2004, Vancouver and Whistler, British Columbia, Canada]*, pp. 321–328, MIT Press, 2004.

[34] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semi-supervised learning on large graphs," in *The 17th Annual Conference on Learning Theory, COLT 2004, Banff, Canada, July 1-4, 2004*, vol. 3120, pp. 624–638, 2004.

[35] D. Zhou and B. Schölkopf, "Regularization on discrete spaces," in *The 27th DAGM Symposium, Vienna, Austria, August 31 - September 2, 2005*, vol. 3663, pp. 361–368, 2005.

[36] A. Weihs and M. Thorpe, "Consistency of fractional graph-laplacian regularization in semi-supervised learning with finite labels," *CoRR*, vol. abs/2303.07818, 2023.

[37] S. G. Krantz and H. R. Parks, *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

[38] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *CoRR*, vol. abs/1903.02428, 2019.

[39] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," *Journal of Complex Networks*, vol. 9, no. 2, 2021.

[40] D. Lim, F. Hohne, X. Li, S. L. Huang, V. Gupta, O. Bhalerao, and S. Lim, "Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual* (M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds.), pp. 20887–20902, 2021.

[41] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008.

[42] M. Zitnik and J. Leskovec, "Predicting multicellular function through multi-layer tissue networks," *Bioinformatics*, vol. 33, no. 14, pp. i190–i198, 2017.

[43] H. Pei, B. Wei, K. C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[44] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 1725–1735, PMLR, 2020.

[45] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), 2020.

[46] P. Yanardag and S. V. N. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015* (L. Cao, C. Zhang, T. Joachims, G. I. Webb, D. D. Margineantu, and G. Williams, eds.), pp. 1365–1374, ACM, 2015.

[47] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[48] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018* (S. A. McIlraith and K. Q. Weinberger, eds.), pp. 4438–4445, AAAI Press, 2018.

[49] C. Gallicchio and A. Micheli, "Fast and deep graph neural networks," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 3898–3905, AAAI Press, 2020.

[50] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[51] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain* (D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, eds.), pp. 1993–2001, 2016.

[52] J. Klicpera, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds.), pp. 13333–13345, 2019.

[53] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel, "Lanczosnet: Multi-scale deep graph convolutional networks," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[54] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 6572–6583, 2018.

[55] P. Kidger, "On neural differential equations," *CoRR*, vol. abs/2202.02435, 2022.

[56] M. Eliasof, E. Haber, and E. Treister, "PDE-GCN: novel architectures for graph neural networks motivated by partial differential equations," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 3836–3849, 2021.

[57] B. Chamberlain, J. Rowbottom, D. Eynard, F. D. Giovanni, X. Dong, and M. M. Bronstein, "Beltrami flow and neural diffusion on graphs," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 1594–1609, 2021.

[58] G. Fu, Y. Hou, J. Zhang, K. Ma, B. F. Kamhoua, and J. Cheng, "Understanding graph neural networks from graph signal denoising perspectives," *CoRR*, vol. abs/2006.04386, 2020.

[59] M. Zhu, X. Wang, C. Shi, H. Ji, and P. Cui, "Interpreting and unifying graph neural networks with an optimization framework," in *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021* (J. Leskovec, M. Grobelnik, M. Najork, J. Tang, and L. Zia, eds.), pp. 1215–1226, ACM / IW3C2, 2021.

[60] Y. Yang, T. Liu, Y. Wang, J. Zhou, Q. Gan, Z. Wei, Z. Zhang, Z. Huang, and D. Wipf, "Graph neural networks inspired by classical iterative algorithms," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 11773–11783, PMLR, 2021.

[61] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Machine Learning, Proceedings of the Twentieth International Conference, ICML 2003, August 21-24, 2003, Washington, DC, USA* (T. Fawcett and N. Mishra, eds.), pp. 912–919, AAAI Press, 2003.

[62] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings* (B. Schölkopf and M. K. Warmuth, eds.), vol. 2777 of *Lecture Notes in Computer Science*, pp. 144–158, Springer, 2003.

[63] B. Nadler, N. Srebro, and X. Zhou, "Semi-supervised learning with the graph Laplacian: The limit of infinite unlabelled data," *Advances in Neural Information Processing Systems, NIPS 2009*, vol. 22, pp. 1330–1338, 2009.

[64] D. Slepcev and M. Thorpe, "Analysis of p-laplacian regularization in semi-supervised learning," *CoRR*, vol. abs/1707.06213, 2017.

[65] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, 2006.

[66] P. Niyogi, "Manifold regularization and semi-supervised learning: some theoretical analyses," *Journal of Machine Learning Research*, vol. 14, no. 1, pp. 1229–1250, 2013.

[67] X. J. Zhu, "Semi-supervised learning literature survey," 2005.

[68] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, no. 2, pp. 373–440, 2020.

[69] F. D. Giovanni, J. Rowbottom, B. P. Chamberlain, T. Markovich, and M. M. Bronstein, "Graph neural networks as gradient flows," *CoRR*, vol. abs/2206.10991, 2022.

[70] S. P. Meyn and R. L. Tweedie, *Markov chains and stochastic stability*. Springer Science & Business Media, 2012.

[71] H. D. Macedo and J. N. Oliveira, "Typing linear algebra: A biproduct-oriented approach," *Science of Computer Programming*, vol. 78, no. 11, pp. 2160–2191, 2013.

[72] F. R. Chung, *Spectral graph theory*, vol. 92. American Mathematical Society, 1997.

[73] M. Hein, J. Audibert, and U. von Luxburg, "Graph laplacians and their convergence on random neighborhood graphs," *Journal of Machine Learning Research*, vol. 8, pp. 1325–1368, 2007.

[74] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," *arXiv preprint arXiv:2007.08663*, 2020.

[75] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

Appendix of Implicit Graph Neural Diffusion Based on Constrained Dirichlet Energy Minimization

Contents

A Related Work and Discussion **15**

B Theorems and Proofs **17**

B.1 Proof of Lemma 1 17

B.2 Proof of Lemma 2 18

B.3 Proof of Theorem 3 18

B.4 Proof of Lemma 4 19

B.5 Proof of Corollary 5 20

B.6 Proof of Lemma 6 21

B.7 Proof of Corollary 7 21

B.8 Proposition 8 and proof 22

B.9 Lemma 9 and proof 23

C Omitted Derivations **24**

C.1 Derivation of parametrized Dirichlet energy 24

C.2 Derivation of the fixed-point equilibrium equation 24

C.3 Derivation of the backward pass gradient computation 25

D Canonical Graph Laplacian Operators and Dirichlet Energies **28**

D.1 Canonical graph Laplcian operators 28

D.2 Canonical Dirichlet energies 29

E Experimental Setup and Additional Experiments **30**

E.1 Experimental setup 30

E.2 Results for the learned graph geometry 30

E.3 Results with different iteration numbers 33

E.4 Running time of DIGNNs 33

A Related Work and Discussion

In this section, we discuss the related work and highlight our contributions against existing approaches. Our work generally pertains to several areas, including implicit GNNs, graph neural diffusions, optimization-inspired explicit GNNs, and graph-based semi-supervised learning.

Implicit GNNs. Instead of stacking a series of propagation layers hierarchically, implicit GNNs define their outputs as solutions to fixed-point equations, which are equivalent to running infinite-depth GNNs. As a result, implicit GNNs are able to capture long-range dependencies and benefit from the global receptive field. Moreover, by using implicit differentiation [37] for the backward pass gradient computation, implicit GNNs require only constant training memory [20, 21]. A series of implicit GNN architectures, e.g., [50, 22, 23, 32, 24, 25], have been proposed. Probably one of the earliest implicit GNN models, proposed by [50], is based on a constrained information diffusion mechanism and it guarantees the existence of a unique stable equilibrium. The implicit layers of IGNN [22] and EIGNN [23] are developed based on the aggregation scheme of GCN [2], which corresponds to an isotropic linear diffusion operator that treats all neighbors equally and is independent of the neighbor features [28, 24]. CGS [32] adopts an input-dependent learnable propagation matrix in their implicit layer which ensures guaranteed convergence and uniqueness for the fixed-point solution. However, [27] shows that isotropic linear diffusion operators would lead to the over-smoothing issue. Therefore, GIND [24] designs a non-linear diffusion operator with anisotropic properties whose equilibrium state corresponds to a convex objective and the anisotropic property may mitigate over-smoothing. Additionally, to incorporate multiscale information on graphs, MGNNI [25] expands the effective range of EIGNN for capturing long-range dependencies to multiscale graph structures.

The underlying geometry of the graph has been shown to have a profound connection with the performance of GNNs in heterophilic settings and their over-smoothing behaviours [30]. However, existing implicit GNNs have mainly concentrated on learning the diffusion strength without explicitly engaging in learning the geometry of vertex and edge spaces. This may limit their adaptability to learning the graph geometry and potentially hinder their performance in graph learning problems. On the contrary, we use three parametrized positive real-valued functions to explicitly learn the Hilbert spaces for vertices, the Hilbert space for edges, and the graph gradient operator, respectively. As a result, our model could have more flexibility to model the graph geometry.

Graph neural diffusions. GNNs can be interpreted as diffusion processes on graphs. [51, 52] design their graph convolutions based on diffusion operators and [53] introduces a polynomial graph filter to exploit multi-scale graph information which is closely related to diffusion maps. On the other hand, neural differential equations [54, 55] have also been applied to design graph neural diffusion processes. Continuous depth GNNs, such as CGNN [31], GRAND [26], PDE-GCN [56], have been developed based on the discretization of diffusion partial differential equations (PDEs) on graphs. [29] introduces a source term into the diffusion process of GRAND to mitigate the over-smoothing issue. [28] shows the equivalence between the propagation scheme of GCN and the numerical discretization of an isotropic diffusion process. Specific PDEs could be helpful to improve the capability of GNNs in capturing the graph geometry or mitigating the over-smoothing issue. For example, [57] develops graph diffusion equations based on the discretized Beltrami flow which facilitates feature learning, positional encoding, and graph rewiring within the diffusion process. [19] employs cellular sheaf theory to develop neural diffusion processes capable of handling heterophilic graphs and addressing the over-smoothing issue.

Similar to existing implicit GNNs, most current graph neural diffusion models also mainly focus on learning the diffusion strength and do not explicitly involve modeling the geometry of vertex and edge spaces, which may restrict their capability to effectively capture the underlying graph geometry. In contrast, our model could have more flexibility to model the graph geometry. Moreover, our model adopts an implicit graph diffusion layer instead of explicit diffusion discretization, which admits an equilibrium state that corresponds to an infinite number of diffusion steps. It enables us to expand the receptive field of diffusion processes without the need for manual adjustment of the terminal time and step size in the diffusion equation. Note also that [29] presents a random walk interpretation of GRAND and shows that the node features will become the same across all nodes under the GRAND-l dynamics (see Proposition 3 in [29]). Their theoretical analysis states that GRAND-l suffers from the over-smoothing during inference (OSI) phenomenon introduced in Lemma 6.

Optimization-inspired explicit GNNs. There have been multiple works, e.g., [58, 59, 60, 10], that establish a connection between some linear propagation schemes and optimization frameworks with Laplacian regularizations. [58] provides a graph signal denoising perspective to show that some spectral GNNs, e.g., ChebyNet [1], GCN [2], SGC [4], work as node feature denoising and smoothing. [59] develops an optimization framework to design adjustable graph convolutional kernels with low-pass or high-pass filtering capabilities. [60] designs GNN architecture based on an iteration algorithm applied to minimize a regularized energy function. [10] derives a p-Laplacian based message passing scheme from a p-Laplacian regularization framework, which can adaptively work as low-pass and low-high-pass filters.

These optimization-inspired GNNs are all limited to explicit layers and they only gather limited hop neighbor information due to limited propagation steps. Moreover, their optimization frameworks are usually developed based on fixed graph structures where the edge features are independent of node features. Different from these methods, our model is able to capture very long-range dependencies and benefit from the global receptive field. We also parameterize the vertex and edge Hilbert spaces as well as the graph gradient operator so that the underlying graph geometry to be learnable.

Graph-based semi-supervised learning. Graph-based semi-supervised learning algorithms usually assume that the labels of neighboring nodes in a graph are the same or consistent. Over the past few decades, numerous methods have been proposed. They generally intend to minimize a smoothness regularizer with label constraints at some vertices to ensure label consistency among neighboring nodes. Such a smoothness regularizer can be various, including graph Laplacian regularization [61, 62, 33, 34, 35, 63, 64, 36], Tikhonov regularization [34], manifold regularizations [65, 66]. We refer to [67, 68] for a more comprehensive review of graph-based semi-supervised learning.

These methods are developed based on non-parameterized graph Laplacians whose vertex and edge spaces are fixed and independent of the node features. As we characterized in Corollary 5, this could lead to *over-smoothing during training (OST)*, i.e., changing the node features will not affect the output node representations during training. Moreover, when it comes to graph classification or graph regression problems which do not have label constraints during the inference process, Lemma 4 shows that their output may converge to a constant function over all nodes regardless of the input features, a problem we call *over-smoothing during inference (OSI)*. Grounding on the theoretical justifications, our derived model can overcome these two types of over-smoothing issues.

Compare to existing optimization views for implicit GNNs and graph neural diffusions. It is worth noting that there are two works [24, 69], that have proposed optimization views for designing implicit GNNs and graph neural diffusion models. [24] shows that there exists a convex objective function whose solution corresponds to the equilibrium of their proposed implicit layer and [69] regards residual GNNs as a gradient flow equation of a parametrized Dirichlet energy functional. There are several key differences between our work and [24, 69]: (1) Our work explicitly parameterizes the vertex and edge Hilbert spaces, providing more flexibility in modeling graph geometry. In contrast, [24, 69] do not involve such explicit parameterizations. (2) We explicitly specify the optimization objective function used to derive the implicit layer. However, [24] only establishes the existence of a convex objective function without explicitly defining it. (3) Our optimization framework theoretically characterizes the conditions for two types of over-smoothing issues. Conversely, [24] does not delve into the theoretical aspects of over-smoothing under their optimization framework. (4) Our optimization framework aims to develop implicit layers with guarantees for a unique equilibrium state and quick convergence. However, [69] focuses on designing explicit graph neural diffusion equations based on the discretization of the gradient flow.

Limitations. One of the main limitations of our work is that our theoretical analysis and methodology have focused on linear graph neural diffusion operators. However, the theoretical foundation for non-linear dynamic systems remains an important impediment to the entire field of machine learning. Nonetheless, our work has provided some valuable insights into the interplay between implicit GNNs, differential geometry, and semi-supervised learning. There are still numerous unexplored insights to be discovered and we look forward to seeing further cross-fertilization between machine learning and algebraic topology in the future.

Societal impact. Due to the theoretical nature of this work, we are not aware of any immediate negative societal impacts.

B Theorems and Proofs

B.1 Proof of Lemma 1

Lemma 1. *Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a function $g : \mathcal{E} \mapsto \mathbb{R}$, the graph divergence* $\mathrm{div} : \mathcal{H}(\mathcal{E}, \phi) \mapsto \mathcal{H}(\mathcal{V}, \chi)$ *is explicitly given by*

$$(\mathrm{div} g)(i) = \frac{1}{2\chi(i)} \sum_{j=1}^{N} \varphi([i,j])\phi([i,j]) \left(g([i,j]) - g([j,i])\right).$$

Proof. For any $k = 1, 2, \ldots, N$, using the indicator function $f(i) = \mathbb{I}_{i=k}$, where

$$\mathbb{I}_{i=k} = \begin{cases} 1 & \text{if } i = k, \\ 0 & \text{if } i \neq k. \end{cases}$$

By the definition of the inner product on the function space on the edges, we have

$$\begin{aligned}
\langle \nabla \mathbb{I}_{\cdot=k}, g \rangle_{\mathcal{E}} &= \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\nabla \mathbb{I}_{\cdot=k})([i,j]) g([i,j]) \phi([i,j]) \\
&= \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \varphi([i,j])(\mathbb{I}_{j=k} - \mathbb{I}_{i=k}) g([i,j]) \phi([i,j]) \\
&= \frac{1}{2} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \varphi([i,j]) \mathbb{I}_{j=k} g([i,j]) \phi([i,j]) - \sum_{i=1}^{N} \sum_{j=1}^{N} \varphi([i,j]) \mathbb{I}_{i=k}) g([i,j]) \phi([i,j]) \right) \\
&= \frac{1}{2} \left(\sum_{i=1}^{N} \varphi([i,k]) g([i,k]) \phi([i,k]) - \sum_{j=1}^{N} \varphi([k,j]) g([k,j]) \phi([k,j]) \right) \\
&= \frac{1}{2} \left(\sum_{j=1}^{N} \varphi([j,k]) g([j,k]) \phi([j,k]) - \sum_{j=1}^{N} \varphi([k,j]) g([k,j]) \phi([k,j]) \right) \\
&\qquad\qquad \text{(Replace } i \text{ by } j \text{ for the first term)} \\
&= \frac{1}{2} \left(\sum_{j=1}^{N} \varphi([k,j]) \phi([k,j]) g([j,k]) - \sum_{j=1}^{N} \varphi([k,j]) \phi([k,j]) g([k,j]) \right) \\
&\qquad\qquad ([k,j] = [j,k] \text{ for undirected graphs}) \\
&= \frac{1}{2} \sum_{j=1}^{N} \varphi([k,j]) \phi([k,j]) \left(g([j,k]) - g([k,j])\right).
\end{aligned}$$

Note that

$$\begin{aligned}
\langle \mathbb{I}_{\cdot=k}, \mathrm{div} g \rangle_{\mathcal{V}} &= \sum_{i=1}^{N} (\mathbb{I}_{i=k})(\mathrm{div} g)(i) \chi(i) \\
&= (\mathrm{div} g)(k) \chi(k).
\end{aligned}$$

Note also that

$$\begin{aligned}
&\langle \mathbb{I}_{\cdot=k}, -\mathrm{div} g \rangle_{\mathcal{V}} = \langle \nabla \mathbb{I}_{\cdot=k}, g \rangle_{\mathcal{E}} \\
\implies &(\mathrm{div} g)(k) \chi(k) = -\frac{1}{2} \sum_{j=1}^{N} \varphi([k,j]) \phi([k,j]) \left(g([j,k]) - g([k,j])\right) \\
\implies &(\mathrm{div} g)(k) = \frac{1}{2\chi(k)} \sum_{j=1}^{N} \varphi([k,j]) \phi([k,j]) \left(g([k,j]) - g([j,k])\right).
\end{aligned}$$

□

B.2 Proof of Lemma 2

Lemma 2. *Given a function $f : \mathcal{V} \mapsto \mathbb{R}$, the graph Laplacian $\Delta : \mathcal{H}(\mathcal{V}, \chi) \mapsto \mathcal{H}(\mathcal{V}, \chi)$ is given by:*

$$(\Delta f)(i) = \frac{1}{\chi(i)} \sum_{j=1}^{N} \varphi([i,j])^2 \phi([i,j]) \left(f(i) - f(j)\right).$$

Proof.

$$\begin{aligned}
(\Delta f)(i) &= (-\mathrm{div}\nabla f)(i) \\
&= -\frac{1}{2\chi(i)} \sum_{j=1}^{N} \varphi([i,j])\phi([i,j]) \left(\nabla f([i,j]) - \nabla f([j,i])\right) \\
&= \frac{1}{2\chi(i)} \sum_{j=1}^{N} \varphi([i,j])\phi([i,j]) \left(\nabla f([j,i]) - \nabla f([i,j])\right) \\
&= \frac{1}{2\chi(i)} \sum_{j=1}^{N} \varphi([i,j])\phi([i,j]) \left(\varphi([j,i]) \left(f(i) - f(j)\right) - \varphi([i,j]) \left(f(j) - f(i)\right)\right) \\
&= \frac{1}{2\chi(i)} \sum_{j=1}^{N} \varphi([i,j])\phi([i,j]) \left(\varphi([i,j]) \left(f(i) - f(j)\right) - \varphi([i,j]) \left(f(j) - f(i)\right)\right) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ([j,i] = [i,j]) \\
&= \frac{1}{\chi(i)} \sum_{j=1}^{N} \varphi([i,j])^2 \phi([i,j]) \left(f(i) - f(j)\right).
\end{aligned}$$

□

B.3 Proof of Theorem 3

Theorem 3 (Spectral Range of Δ_Φ). *Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ with node embeddings $\mathbf{X} \in \mathbb{R}^{N \times d}$, let positive real-valued functions χ, ϕ, φ of the graph neural Laplacian Δ_Φ be parameterized by Equations (8) to (10), respectively. Suppose that $\|\mathbf{x}_i\| \leq c_X$ for all $i \in \mathcal{V}$ and the parameters Θ_χ, Θ_ϕ can be bounded by $\|\Theta_\chi\| \leq c_\chi, \|\Theta_\phi\| \leq c_\phi$ for some $c_X, c_\chi, c_\phi \in \mathbb{R}_+^*$. If $\mathcal{G}$ is connected and λ is a eigenvalue associated with the eigenvector $\mathbf{u}$ of Δ_Φ, then*

$$0 \leq \lambda \leq 2c_\phi^2 c_\chi c_X \cosh(c_\chi c_X).$$

Proof. Note that by Proposition 8 the graph neural Laplacian is positive semi-definite and therefore we have $\lambda \geq 0$. On the other hand, by the definition of graph Laplacian, we have for all $i = 1, 2, \ldots, N$,

$$(\Delta \mathbf{u})_i = \frac{1}{\chi(i)} \sum_{j=1}^{N} \varphi([i,j])^2 \phi([i,j])(u_i - u_j) = \lambda u_i.$$

Then we have for all $i = 1, \ldots, N$,

$$
\begin{aligned}
\lambda &= \frac{1}{u_i} \sum_{j=1}^{N} \frac{\varphi([i,j])^2 \phi([i,j])}{\chi(i)} (u_i - u_j) \\
&= \sum_{j=1}^{N} \frac{\varphi([i,j])^2 \phi([i,j])}{\chi(i)} (1 - \frac{u_j}{u_i}) \\
&\leq \sum_{j=1}^{N} \frac{\varphi([i,j])^2 \phi([i,j])}{\chi(i)} \left(1 + \left|\frac{u_j}{u_i}\right|\right) \\
&= \sum_{j=1}^{N} \frac{A_{i,j}}{D_i} \frac{\tanh((\|\Theta_\varphi(\mathbf{x}_i - \mathbf{x}_j)\| + \epsilon)^{-1}) \tanh(|(\Theta_\phi \Theta_\chi \mathbf{x}_i)^\top (\Theta_\phi \Theta_\chi \mathbf{x}_j)|)}{\tanh(\|\Theta_\chi \mathbf{x}_i\|)} \left(1 + \left|\frac{u_j}{u_i}\right|\right) \\
&\leq \sum_{j=1}^{N} \frac{A_{i,j}}{D_i} \frac{\tanh(|(\Theta_\phi \Theta_\chi \mathbf{x}_i)^\top (\Theta_\phi \Theta_\chi \mathbf{x}_j)|)}{\tanh(\|\Theta_\chi \mathbf{x}_i\|)} \left(1 + \left|\frac{u_j}{u_i}\right|\right) && (\tanh(a) \leq 1, \forall a \in \mathbb{R}) \\
&\leq \sum_{j=1}^{N} \frac{A_{i,j}}{D_i} \frac{|(\Theta_\phi \Theta_\chi \mathbf{x}_i)^\top (\Theta_\phi \Theta_\chi \mathbf{x}_j)|}{\tanh(\|\Theta_\chi \mathbf{x}_i\|)} \left(1 + \left|\frac{u_j}{u_i}\right|\right) && (\tanh(a) \leq a, \forall a \in \mathbb{R}) \\
&= \sum_{j=1}^{N} \frac{A_{i,j}}{D_i} \frac{|(\Theta_\phi \Theta_\chi \mathbf{x}_i)^\top (\Theta_\phi \Theta_\chi \mathbf{x}_j)| \cosh(\|\Theta_\chi \mathbf{x}_i\|)}{\sinh(\|\Theta_\chi \mathbf{x}_i\|)} \left(1 + \left|\frac{u_j}{u_i}\right|\right) \\
&\leq \sum_{j=1}^{N} \frac{A_{i,j}}{D_i} \frac{\|\Theta_\phi\| \|\Theta_\chi \mathbf{x}_i\| \|\Theta_\phi \Theta_\chi \mathbf{x}_j\| \cosh(\|\Theta_\chi \mathbf{x}_i\|)}{\sinh(\|\Theta_\chi \mathbf{x}_i\|)} \left(1 + \left|\frac{u_j}{u_i}\right|\right) \\
&\leq \sum_{j=1}^{N} \frac{A_{i,j}}{D_i} \|\Theta_\phi\| \|\Theta_\phi \Theta_\chi \mathbf{x}_j\| \cosh(\|\Theta_\chi \mathbf{x}_i\|) \left(1 + \left|\frac{u_j}{u_i}\right|\right) && (\tfrac{a}{\sinh(a)} \leq 1, \forall a \in \mathbb{R}) \\
&\leq \sum_{j=1}^{N} \frac{A_{i,j}}{D_i} \|\Theta_\phi\|^2 \|\Theta_\chi\| \|\mathbf{x}_j\| \cosh(\|\Theta_\chi\| \|\mathbf{x}_i\|) \left(1 + \left|\frac{u_j}{u_i}\right|\right) \\
&\leq c_\phi^2 c_\chi c_X \beta \cosh(c_\chi c_X) \left(\sum_{j=1}^{N} \frac{A_{i,j}}{D_i} \left(1 + \left|\frac{u_j}{u_i}\right|\right)\right).
\end{aligned}
$$

Since the above inequality holds for all $i = 1, \ldots, N$, let $k = \arg\max_i(\{|u_i|\}_{i=1,2,\ldots,N})$, we have

$$
\begin{aligned}
\lambda &\leq c_\phi^2 c_\chi c_X \cosh(c_\chi c_X) \left(\sum_{j=1}^{N} \frac{A_{k,j}}{D_k} \left(1 + \left|\frac{u_j}{u_k}\right|\right)\right) \\
&\leq c_\phi^2 c_\chi c_X \cosh(c_\chi c_X) \left(\sum_{j=1}^{N} \frac{A_{k,j}}{D_k} (1 + 1)\right) \\
&= 2 c_\phi^2 c_\chi c_X \cosh(c_\chi c_X).
\end{aligned}
$$

□

B.4 Proof of Lemma 4

Lemma 4 (Convergence and Uniqueness Analysis). *Let γ_{max} be the largest singular value of $\mathbf{C}$. For all $i \in \mathcal{V}$, the fixed-point equilibrium equation $f = \mathbf{Y}' + \mathbf{C} f$ has a unique solution if $\gamma_{max} < 1$. The solution can be obtained by iterating the equation:*

$$
f^{(t+1)} = \mathbf{Y}' + \mathbf{C} f^{(t)}, \quad t = 0, 1, \ldots.
$$

to obtain $f^\star = \lim_{t \to \infty} f^{(t)}$ which is independent of the initial state $f^{(0)}$.

Proof. For $t \geq 1$ and the sequence $f^{(0)}, f^{(1)}, \dots$ generated by $f^{(t+1)} = \mathbf{Y}' + \mathbf{C}f^{(t)}$,

$$\begin{aligned}
\|f^{(t+1)} - f^{(t)}\| &\leq \left\|\mathbf{Y}' + \mathbf{C}f^{(t)} - \mathbf{Y}' - \mathbf{C}f^{(t-1)}\right\| \\
&= \left\|(\mathbf{C}f^{(t)} - \mathbf{C}f^{(t-1)}\right\| \\
&\leq \|\mathbf{C}\|\|f^{(t)} - f^{(t-1)}\|.
\end{aligned}$$

For $n > m \geq 1$, the following inequality holds,

$$\begin{aligned}
\|f^{(n)} - f^{(m)}\| &= \|f^{(n)} - f^{(n-1)} + f^{(n-1)} - \dots + f^{(m+1)} - f^{(m)}\| \\
&\leq \sum_{t=m}^{n-1} \|f^{(t+1)} - f^{(t)}\| \\
&\leq \sum_{t=m}^{n-1} \|\mathbf{C}\|^{t-m}\|f^{(m+1)} - f^{(m)}\| \\
&\leq \sum_{t=m}^{n-1} \|\mathbf{C}\|^{t-m}\|\mathbf{C}\|^{m}\|f^{(1)} - f^{(0)}\| \\
&= \|\mathbf{C}\|^{m} \sum_{t=0}^{n-m-1} \|\mathbf{C}\|^{t}\|f^{(1)} - f^{(0)}\| \\
&\leq \|\mathbf{C}\|^{m} \sum_{t=0}^{\infty} \|\mathbf{C}\|^{t}\|f^{(1)} - f^{(0)}\| \\
&= \|\mathbf{C}\|^{m}(1 - \|\mathbf{C}\|)^{-1}\|f^{(1)} - f^{(0)}\|.
\end{aligned}$$

If $\gamma_{\text{max}} < 1$, we have $\|\mathbf{C}\| = \gamma_{\text{max}} < 1$, which implies that $\|\mathbf{C}\|^{m}(1 - \|\mathbf{C}\|)^{-1}\|f^{(1)} - f^{(0)}\| \geq 0$, then the inequality $\|f^{(n)} - f^{(m)}\| \leq (\|\mathbf{C}\|)^{m}(1 - \|\mathbf{C}\|)^{-1}\|f^{(1)} - f^{(0)}\|$ is well-posed. Note also that when $\|\mathbf{C}\| < 1$, we have $\lim_{t\to\infty}(\|\mathbf{C}\|)^{t} = 0$ and $\lim_{t\to\infty}(\|\mathbf{C}\|)^{t}(1 - \|\mathbf{C}\|)^{-1} = 0$. Then the sequence $f^{(0)}, f^{(1)}, f^{(2)}, \dots$ is a Cauchy sequence because

$$0 \leq \lim_{m\to\infty} \|f^{(n)} - f^{(m)}\| \leq \lim_{m\to\infty} (\|\mathbf{C}\|)^{m}(1 - \|\mathbf{C}\|)^{-1}\|f^{(1)} - f^{(0)}\| = 0.$$

Thus the sequence converges to some solution of $f^{\star} = \mathbf{Y}' + \mathbf{C}f^{\star}$.

To prove the uniqueness, suppose that both $f^{(a)}$ and $f^{(b)}$ satisfy $f = \mathbf{Y}' + \mathbf{C}f$, i.e., $f^{(a)} = \mathbf{Y}' + \mathbf{C}f^{(a)}$ and $f^{(b)} = \mathbf{Y}' + \mathbf{C}f^{(b)}$. Then we have

$$\begin{aligned}
0 &\leq \|f^{(a)} - f^{(b)}\| \\
&= \left\|\mathbf{Y}' + \mathbf{C}f^{(a)} - \mathbf{Y}'(i) - \mathbf{C}f^{(b)}\right\| \\
&= \left\|\mathbf{C}f^{(a)} - \mathbf{C}f^{(b)}\right\| \\
&\leq \|\mathbf{C}\|\|f^{(a)} - f^{(b)}\| \\
&= \|\mathbf{C}\| \left\|\mathbf{Y}' + \mathbf{C}f^{(a)} - \mathbf{Y}' - \mathbf{C}f^{(b)}\right\| \\
&\leq \lim_{t\to\infty} (\|\mathbf{C}\|)^{t} \|f^{(a)} - f^{(b)}\| = 0, \qquad (\lim_{t\to\infty} (\|\mathbf{C}\|)^{t} = 0 \text{ when } \gamma_{\text{max}} < 1)
\end{aligned}$$

which indicates that $f^{(a)} = f^{(b)}$ and there exists unique solution to $f = \mathbf{Y}' + \mathbf{C}f$ if $\gamma_{\text{max}} < 1$.

□

B.5 Proof of Corollary 5

Corollary 5 (Over-Smoothing during Training (OST) Condition). *Assume that the initial state $f^{(0)}$ at node i is initialized as a function of $\mathbf{x}_i$. Assume further that χ, ϕ, φ are not functions of $\mathbf{x}_i$, e.g., Δ is the unnormalized Laplacian or random walk Laplacian. Let γ_{max} be the largest singular value of $\mathbf{C}$. Then, if $\gamma_{max} < 1$, the equilibrium of the fixed point equation Eq. (14) is independent of the node features $\mathbf{x}_i$, i.e. the same equilibrium solution is obtained even if the node features $\mathbf{x}_i$ are changed.*

Proof. When $\gamma_{\max} < 1$, by Lemma 4 we know that the fixed-point equation Eq. (14) has a unique equilibrium. Since χ, φ, and φ are not functions of $\mathbf{x}_i$, Δ is independent of the node features $\mathbf{x}_i, \forall i \in \mathcal{V}$. Therefore, $\gamma_{\max}$ is independent of the node features, which indicates that the uniqueness condition of the equilibrium is also independent of the node features. □

B.6 Proof of Lemma 6

Lemma 6 (Over-Smoothing during Inference (OSI) Condition). *When $\mathcal{I} = \emptyset$, i.e., there is no constrained nodes in Eq. (13), we have $\mathbf{C} = \mathbf{P}$ and $\mathbf{y}'(i) = \mathbf{0}, \forall i \in \mathcal{V}$. $\mathbf{P}$ is a Markov matrix and the solution of Eq. (13) satisfies $f = \mathbf{P}f$. If the graph $\mathcal{G}$ is connected and non-bipartite, i,e., the Markov chain corresponding to $\mathbf{P}$ is irreducible and aperiodic, the equilibrium $f^\star(i)$ for all nodes are identical and not unique, i.e., $f^\star(i) = \mathbf{v}, \forall i \in \mathcal{V}$, where $\mathbf{v}$ is any vector in $\mathbb{R}^c$ and is independent of the labels. Iterating $f^{(t+1)} = \mathbf{P}f^{(t)}, t = 0, 1, \ldots$, with initial state $f^{(0)} \in \mathbb{R}^{N\times c}$ will converge to*

$$f^\star(i) = (\lim_{t\to\infty} \mathbf{P}^t f^{(0)})(i) = (\boldsymbol{\pi} f^{(0)})^\top, \quad \textit{for all } i \in \mathcal{V},$$

where $\boldsymbol{\pi} = (\widehat{D}_1/\sum_j \widehat{D}_j, \ldots, \widehat{D}_N/\sum_j \widehat{D}_j) \in \mathbb{R}^{1\times N}$ is the stationary distribution of $\mathbf{P}$.

Proof. If the graph is connected and non-bipartite, the Markov chain corresponding to $\mathbf{P}$ is irreducible and aperiodic. Then by the Fundamental Theorem of Markov chains [70], $\mathbf{P}$ has a unique stationary distribution. Let $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N) \in \mathbb{R}^{1\times N}$ be the stationary distribution of $\mathbf{P}$, then

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}.$$

$\boldsymbol{\pi}$ is given by $(\widehat{D}_1/\sum_j \widehat{D}_j, \ldots, \widehat{D}_N/\sum_j \widehat{D}_j)$, which can be easily verified by

$$\begin{aligned}
\sum_{i=1}^N \pi_i P_{i,j} &= \sum_{i=1}^N \frac{\widehat{D}_i}{\sum_{k=1}^N \widehat{D}_k} \frac{\varphi([i,j])^2 \phi([i,j])}{\widehat{D}_i} \\
&= \frac{\sum_{i=1}^N \varphi([i,j])^2 \phi([i,j])}{\sum_{k=1}^N \widehat{D}_k} \\
&= \frac{\widehat{D}_j}{\sum_{k=1}^N \widehat{D}_k} \\
&= \pi_j.
\end{aligned}$$

By the Fundamental Theorem of Markov chains, we have

$$\lim_{t\to\infty} (\mathbf{P}^t)_{i,j} = \pi_j, \quad \text{for all } i, j \in \mathcal{V},$$

which indicates that as t goes to infinity, all row vectors in $\mathbf{P}^t$ are identical and is the stationary vector $\boldsymbol{\pi}$ of $\mathbf{P}$, i.e., $\lim_{t\to\infty} \mathbf{P}^t = (\boldsymbol{\pi}^\top, \ldots, \boldsymbol{\pi}^\top)^\top$. Denote $f^\star = (f^\star(1), \ldots, f^\star(N))^\top \in \mathbb{R}^{N\times c}$, then

$$f^\star(i) = (\mathbf{P}f^\star)(i) = (\mathbf{P}\mathbf{P}f^\star)(i) = (\lim_{t\to\infty} \mathbf{P}^t f^\star)(i) = ((\boldsymbol{\pi}^\top, \ldots, \boldsymbol{\pi}^\top)^\top f)(i) = (\boldsymbol{\pi} f^\star)^\top, \quad \forall i \in \mathcal{V}.$$

which indicates that the equilibrium of all nodes are identical. $(\boldsymbol{\pi} f)^\top$ can be any vector in $\mathbb{R}^c$, therefore any vector $\mathbf{v} \in \mathbb{R}^c$, $f^\star = (\mathbf{v}, \ldots, \mathbf{v})^\top \in \mathbb{R}^{N\times d}$ is an equilibrium of the fixed-point equation $f = \mathbf{P}f$. Hence, the equilibrium is a constant function over all nodes.

Moreover, iterating $f^{(t+1)} = \mathbf{P}f^{(t)}$ with an initial state $f^{(0)}$ with converge to

$$\lim_{t\to\infty} f^{(t+1)}(i) = (\lim_{t\to\infty} \mathbf{P}^t f^{(0)})(i) = ((\boldsymbol{\pi}^\top, \ldots, \boldsymbol{\pi}^\top)^\top f^{(0)})(i) = (\boldsymbol{\pi} f^{(0)})^\top, \quad \text{for all } i \in \mathcal{V},$$

which is an equilibrium of the fixed-point equation $f = \mathbf{P}f$. □

B.7 Proof of Corollary 7

Corollary 7 (Tractable Well-Posedness Condition and Convergence Rate). *Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ with node embeddings $\widetilde{\mathbf{X}} = (\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_N)^\top \in \mathbb{R}^{N\times d}$, Let λ_{max} be the largest*

eigenvalues of the matrix Δ. Suppose that $\mathcal{G}$ is connected, the fixed-point equilibrium equation $\mathbf{Z} = \widetilde{\mathbf{X}} - \frac{1}{\mu}\Delta\mathbf{Z}$ has a unique solution if $\mu > \lambda_{max}$. The solution can be obtained by iterating:

$$\mathbf{Z}^{(t+1)} = \widetilde{\mathbf{X}} - \frac{1}{\mu}\Delta\mathbf{Z}^{(t)}, \quad \textit{with } \mathbf{Z}^{(0)} = \mathbf{0}, \quad t = 0, 1, \ldots.$$

Therefore, $\mathbf{Z}^\star = \lim_{t\to\infty} \mathbf{Z}^{(t)}$. Suppose that $\|\mathbf{Z}^\star\| \leq c_Z \in \mathbb{R}_+^*$, then $\forall t \geq 1$, $\|\mathbf{Z}^{(t)} - \mathbf{Z}^\star\| \leq c_Z \left(\lambda_{max}/\mu\right)^t$. Specifically,

1. *When Δ is the random walk Laplacian, i.e., $\Delta = \Delta^{(rw)}$, $\mu > 2$ and $\|\mathbf{Z}^{(t)} - \mathbf{Z}^\star\| \leq c_Z \left(2/\mu\right)^t$;*
2. *When Δ is learnable and $\Delta = \Delta_\Phi$ as given in Eq. (11), suppose that $\|\widetilde{\mathbf{x}}_i\| \leq c_X$ for all $i \in \mathcal{V}$, the parameters Θ_χ, Θ_ϕ can be bounded by $\|\Theta_\chi\| \leq c_\chi, \|\Theta_\phi\| \leq c_\phi$ and $c_X, c_\chi, c_\phi \in \mathbb{R}_+^*$, we have $\mu > 2c_\phi^2 c_\chi c_X \cosh(c_\chi c_X)$ and $\|\mathbf{Z}^{(t)} - \mathbf{Z}^\star\| \leq c_Z \left(2c_\phi^2 c_\chi c_X \cosh(c_\chi c_X)/\mu\right)^t$.*

Proof. The fixed-point equilibrium equation $\mathbf{Z} = \widetilde{\mathbf{X}} - \frac{1}{\mu}\Delta\mathbf{Z}$ is a case of $f = \mathbf{Y}' + \mathbf{C}f$ when $\mathcal{I} = \mathcal{V}$. Then we have $\mathbf{C} = -\frac{1}{\mu}\Delta$. By Proposition 8, Δ is positive semi-definite. Therefore, the largest singular value of Δ is its largest eigenvalue. Then we have

$$\|\mathbf{C}\| = \|\frac{1}{\mu}\Delta\| = \frac{1}{\mu}\|\Delta\| = \frac{1}{\mu}\max_i |\lambda_i| = \frac{1}{\mu}\lambda_{\max},$$

which indicates that $\gamma = \frac{1}{\mu}\lambda_{\max}$ Therefore, by Lemma 4, we have if $\frac{1}{\mu}\lambda_{\max} < 1$, i.e., $\mu > \lambda_{\max}$, the fixed-point equilibrium equation has a unique solution and iterating Corollary 7 will convergence to the solution, i.e., $\mathbf{Z}^\star = \lim_{t\to\infty} \mathbf{Z}^{(t)}$. Moreover, for $t \geq 1$

$$\begin{aligned}
\|\mathbf{Z}^{(t)} - \mathbf{Z}^\star\| &= \|\widetilde{\mathbf{X}} - \frac{1}{\mu}\Delta\mathbf{Z}^{(t-1)} - \widetilde{\mathbf{X}} + \frac{1}{\mu}\Delta\mathbf{Z}^\star\| \\
&= \frac{1}{\mu}\|\Delta(\mathbf{Z}^{(t-1)} - \mathbf{Z}^\star)\| \\
&\leq \frac{1}{\mu}\|\Delta\|\|\mathbf{Z}^{(t-1)} - \mathbf{Z}^\star\| \\
&\leq \left(\frac{1}{\mu}\|\Delta\|\right)^t \|\mathbf{Z}^{(0)} - \mathbf{Z}^\star\| \\
&= \left(\frac{\lambda_{\max}}{\mu}\right)^t \|\mathbf{Z}^\star\| \\
&\leq c_Z \left(\frac{\lambda_{\max}}{\mu}\right)^t.
\end{aligned}$$

1. When Δ is fixed and $\Delta = \Delta^{(\mathbf{rw})}$. The largest eigenvalue of the random walk matrix $\lambda_{\max} = 2$. Therefore, we have $\mu > 2$ to ensure the uniqueness of the fixed-point equation and the convergence of Corollary 7.

2. When Δ is parameterized and $\Delta = \Delta_\Phi$. With $\|\widetilde{\mathbf{x}}_i\| \leq c_X, \forall i \in \mathcal{V}, \|\Theta_\chi\| \leq c_\chi, \|\Theta_\phi\| \leq c_\phi$, the largest eigenvalue of Δ_Φ is upper-bounded by $\lambda_{\max} \leq 2c_\phi^2 c_\chi c_X \cosh(c_\chi c_X)$ by Theorem 3. Therefore, to ensure the uniqueness of the fixed-point equation and the convergence of Corollary 7 we need $2c_\phi^2 c_\chi c_X \cosh(c_\chi c_X) < \mu$. □

B.8 Proposition 8 and proof

Proposition 8. *For any positive real-valued functions $\chi : \mathcal{V} \mapsto \mathbb{R}_+, \phi : \mathcal{E} \mapsto \mathbb{R}^+, \varphi : \mathcal{E} \mapsto \mathbb{R}_+^*$, the graph Laplacian $\Delta : \mathcal{H}(\mathcal{V}, \chi) \mapsto \mathcal{H}(\mathcal{V}, \chi)$ is self-adjoint and positive semi-definite.*

Proof. By the definition of energy-based graph Laplacian,

$$\begin{aligned}\langle f, \Delta g\rangle_{\mathcal{V}} &= \langle f, -\mathrm{div}\nabla g\rangle_{\mathcal{V}} \\ &= \langle \nabla f, \nabla g\rangle_{\mathcal{E}} \\ &= \langle -\mathrm{div}\nabla f, g\rangle_{\mathcal{V}} \\ &= \langle \Delta f, g\rangle_{\mathcal{V}},\end{aligned}$$

which indicates the self-adjoint and

$$\begin{aligned}\langle f, \Delta f\rangle_{\mathcal{V}} &= \langle f, -\mathrm{div}\nabla f\rangle_{\mathcal{V}} \\ &= \langle \nabla f, \nabla f\rangle_{\mathcal{E}} \geq 0,\end{aligned}$$

which indicates the positive semi-definite. □

B.9 Lemma 9 and proof

Lemma 9. *Given a vector function $f : \mathcal{V} \mapsto \mathbb{R}^c$, for any positive real-valued functions $\chi : \mathcal{V} \mapsto \mathbb{R}_+, \phi : \mathcal{E} \mapsto \mathbb{R}_+, \varphi : \mathcal{E} \mapsto \mathbb{R}_+^*$, we have*

$$\left.\frac{\mathrm{d}\mathcal{S}(f)}{\mathrm{d}f}\right|_i = 2\chi(i)(\Delta f)(i). \tag{20}$$

Proof.

$$\begin{aligned}\left.\frac{\mathrm{d}\mathcal{S}(f)}{\mathrm{d}f}\right|_i &= \left.\frac{\mathrm{d}}{\mathrm{d}f}\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\varphi([i,j])^2\phi([i,j])\left\|f(j)-f(i)\right\|^2\right|_i \\ &= \sum_{i=1}^{N}\varphi([i,j])^2\phi([i,j])(f(j)-f(i)) + \sum_{j=1}^{N}\varphi([i,j])^2\phi([i,j])(f(i)-f(j)) \\ &= \sum_{j=1}^{N}\varphi([j,i])^2\phi([j,i])(f(i)-f(j)) + \sum_{j=1}^{N}\varphi([i,j])^2\phi([i,j])(f(i)-f(j)) \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(exchange } i \text{ and } j \text{ for the first term)} \\ &= 2\sum_{j=1}^{N}\varphi([i,j])^2\phi([i,j])(f(i)-f(j)) \qquad\qquad ([j,i]=[i,j]) \\ &= 2\chi(i)(\Delta f)(i).\end{aligned}$$

□

C Omitted Derivations

C.1 Derivation of parametrized Dirichlet energy

$$
\begin{aligned}
\mathcal{S}(f) &= \|\nabla f\|_{\mathcal{E}}^2 \\
&= \langle \nabla f, \nabla f \rangle_{\mathcal{E}} \\
&= \langle f, \Delta f \rangle_{\mathcal{V}} \\
&= \sum_{i=1}^{N} \langle f(i), (\Delta f)(i) \rangle \, \chi(i) \\
&= \sum_{i=1}^{N} \left\langle f(i), \frac{1}{\chi(i)} \sum_{j=1}^{N} \varphi([i,j])^2 \phi([i,j]) \, (f(i) - f(j)) \right\rangle \chi(i) \\
&= \sum_{i=1}^{N} \sum_{j=1}^{N} \varphi([i,j])^2 \phi([i,j]) \, \langle f(i), f(i) - f(j) \rangle \\
&= \frac{1}{2} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \varphi([i,j])^2 \phi([i,j]) \langle f(i), f(i) - f(j) \rangle + \sum_{i=1}^{N} \sum_{j=1}^{N} \varphi([i,j])^2 \phi([i,j]) \langle f(i), f(i) - f(j) \rangle \right) \\
&= \frac{1}{2} \left(\sum_{j=1}^{N} \sum_{i=1}^{N} \varphi([j,i])^2 \phi([j,i]) \langle f(j), f(j) - f(i) \rangle + \sum_{i=1}^{N} \sum_{j=1}^{N} \varphi([i,j])^2 \phi([i,j]) f(i) \, (f(i) - f(j)) \right) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(exchange } i \text{ and } j \text{ for the first term)} \\
&= \frac{1}{2} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \varphi([i,j])^2 \phi([i,j]) \langle f(j), f(j) - f(i) \rangle - \sum_{i=1}^{N} \sum_{j=1}^{N} \varphi([i,j])^2 \phi([i,j]) \langle f(i), f(j) - f(i) \rangle \right) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ([j,i] = [i,j]) \\
&= \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \varphi([i,j])^2 \phi([i,j]) \langle f(j) - f(i), f(j) - f(i) \rangle \\
&= \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \varphi([i,j])^2 \phi([i,j]) \|f(j) - f(i)\|^2.
\end{aligned}
$$

C.2 Derivation of the fixed-point equilibrium equation

Recall the objective function:

$$
\inf_f \mathcal{L}(f) = \inf_f \left[\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \varphi([i,j])^2 \phi([i,j]) \|f(j) - f(i)\|^2 + \mu \sum_{i \in \mathcal{I}} \|f(i) - \mathbf{y}(i)\|_{\mathcal{V}}^2 \right].
$$

By Lemma 2 we have

$$
\left. \frac{\mathrm{d}\mathcal{S}(f)}{\mathrm{d}f} \right|_i = 2\chi(i)(\Delta f)(i).
$$

Note also that for $i \in \mathcal{I}$,

$$
\begin{aligned}
\left. \frac{\mathrm{d} \sum_{i \in \mathcal{I}} \|f(i) - \mathbf{y}(i)\|_{\mathcal{V}}^2}{\mathrm{d}f} \right|_i &= \left. \frac{\mathrm{d} \sum_{i \in \mathcal{I}} \langle f(i) - \mathbf{y}(i), f(i) - \mathbf{y}(i) \rangle_{\mathcal{V}}}{\mathrm{d}f} \right|_i \\
&= \left. \frac{\mathrm{d} \sum_{i \in \mathcal{I}} \|f(i) - \mathbf{y}(i)\|^2 \chi(i)}{\mathrm{d}f} \right|_i \\
&= 2(f(i) - \mathbf{y}(i))\chi(i). \qquad (21)
\end{aligned}
$$

For labeled nodes $i \in \mathcal{I}$, we have

$$\begin{aligned}\left.\frac{\mathrm{d}\mathcal{L}(f)}{\mathrm{d}f}\right|_i &= \left.\frac{\mathrm{d}\mathcal{S}(f)}{\mathrm{d}f}\right|_i + 2\mu\chi(i)(f(i)-\mathbf{y}(i)) \\ &= 2\chi(i)(\Delta f)(i) + 2\mu\chi(i)(f(i)-\mathbf{y}(i)).\end{aligned} \tag{22}$$

$$\begin{aligned}\left.\frac{\mathrm{d}\mathcal{L}(f)}{\mathrm{d}f}\right|_i = 0 &\Longrightarrow 2\chi(i)(\Delta f^\star)(i) + 2\mu\chi(i)(f^\star(i)-\mathbf{y}(i)) = 0 \\ &\Longrightarrow f^\star(i) = \mathbf{y}(i) - \frac{1}{\mu}(\Delta f^\star)(i).\end{aligned} \tag{23}$$

For unlabeled nodes $i \notin \mathcal{I}$,

$$\left.\frac{\mathrm{d}\mathcal{L}(f)}{\mathrm{d}f}\right|_i = \left.\frac{\mathrm{d}\mathcal{S}(f)}{\mathrm{d}f}\right|_i = 2\chi(i)(\Delta f)(i). \tag{24}$$

$$\begin{aligned}\left.\frac{\mathrm{d}\mathcal{L}(f)}{\mathrm{d}f}\right|_i = 0 \Longrightarrow 2\chi(i)(\Delta f^\star)(i) &= 2\sum_{j=1}^{N}\varphi([i,j])^2\phi([i,j])\left(f^\star(i)-f^\star(j)\right) \\ &= 2\sum_{j=1}^{N}\varphi([i,j])^2\phi([i,j])f^\star(i) - 2\sum_{j=1}^{N}\varphi([i,j])^2\phi([i,j])f^\star(j) \\ &= 0.\end{aligned} \tag{25}$$

$$\begin{aligned}&\Longrightarrow \sum_{j=1}^{N}\varphi([i,j])^2\phi([i,j])f^\star(i) = \sum_{j=1}^{N}\varphi([i,j])^2\phi([i,j])f^\star(j) \\ &\Longrightarrow f^\star(i) = \sum_{j=1}^{N}\frac{\varphi([i,j])^2\phi([i,j])}{\sum_{k=1}^{N}\varphi([i,k])^2\phi([i,k])}f^\star(j) = (\mathbf{P}f^\star)(i)\end{aligned} \tag{26}$$

Because for all $i \in \mathcal{V}$,

$$\mathbf{C}_{i,:} = (1-\delta_i)\mathbf{P}_{i,:} - \delta_i\frac{1}{\mu}\Delta_{i,:} = \begin{cases} -\frac{1}{\mu}\Delta_{i,:} & \text{if } i \in \mathcal{I}, \\ \mathbf{P}_{i,:} & \text{otherwise.}\end{cases} \tag{27}$$

we have $f(i) = \mathbf{y}'(i) + (\mathbf{C}f)(i)$ for all $i \in \mathcal{V}$, i.e.,

$$f = \mathbf{Y}' + \mathbf{C}f. \tag{28}$$

C.3 Derivation of the backward pass gradient computation

In this subsection, we provide the derivation of the gradients of trainable parameters with implicit differentiation. We first introduce the matrix vectorization and the Kronecker product.

C.3.1 Matrix vectorization

For a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, we define its vectorization $\mathrm{vec}(\mathbf{A})$ as

$$\mathrm{vec}(\mathbf{A}) = (A_{1,1},\ldots,A_{m,1},A_{1,2},\ldots,A_{m,2},\ldots,A_{1,n},\ldots,A_{m,n})^\top.$$

C.3.2 Kronecker product

For two matrices $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{p\times q}$, the Kronecker product $\mathbf{A}\otimes\mathbf{B} \in \mathbb{R}^{pm\times qn}$ is defined as

$$\mathbf{A}\otimes\mathbf{B} = \begin{bmatrix} A_{1,1}\mathbf{B} & \ldots & A_{1,n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ A_{m,1}\mathbf{B} & \ldots & A_{m,n}\mathbf{B}\end{bmatrix}.$$

By the definition of the Kronecker product, we have the following property of the vectorization with the Kronecker product [71]:

$$\mathrm{vec}(\mathbf{AB}) = (\mathbf{I}_m\otimes\mathbf{A})\mathrm{vec}(\mathbf{B}).$$

C.3.3 Backward pass gradient computation by implicit differentiation

Denote the loss function by

$$\ell = \ell(\mathbf{Y}, \widehat{\mathbf{Y}}) = \ell(\mathbf{Y}, h_{\Theta^{(2)}}(\mathbf{Z}^\star)).$$

Writing the fixed-point equation Eq. (18) into the vectorization form:

$$\begin{aligned}
\mathrm{vec}(\mathbf{Z}^\star) &= \mathrm{vec}(\mathbf{X} - \frac{1}{\mu}\Delta\mathbf{Z}^\star) \\
&= \mathrm{vec}(\mathbf{X}) - \frac{1}{\mu}\mathrm{vec}(\Delta\mathbf{Z}^\star) \\
&= \mathrm{vec}(\mathbf{X}) - \frac{1}{\mu}(\mathbf{I}\otimes\Delta)\mathrm{vec}(\mathbf{Z}^\star). \qquad (29)
\end{aligned}$$

For simplicity, denote by $\vec{\mathbf{X}} := \mathrm{vec}(\mathbf{X})$ and $\vec{\mathbf{Z}}^\star := \mathrm{vec}(\mathbf{Z}^\star)$. Then Eq. (29) becomes

$$\vec{\mathbf{Z}}^\star = \vec{\mathbf{X}} - \frac{1}{\mu}(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star.$$

By the chain rule, we have

$$\frac{\partial\ell}{\partial(\cdot)} = \frac{\partial\ell}{\partial\vec{\mathbf{Z}}^\star}\frac{\mathrm{d}\vec{\mathbf{Z}}^\star}{\mathrm{d}(\cdot)} = \frac{\partial\ell}{\partial h_\Theta}\frac{\partial h_\Theta}{\partial\vec{\mathbf{Z}}^\star}\frac{\mathrm{d}\vec{\mathbf{Z}}^\star}{\mathrm{d}(\cdot)}$$

Since $\vec{\mathbf{Z}}^*$ and $(\cdot)$ are implicitly related, $\frac{\mathrm{d}\vec{\mathbf{Z}}^\star}{\mathrm{d}(\cdot)}$ cannot be directly obtained by automatic differentiation packages. By differentiating both sides of the equilibrium equation $\vec{\mathbf{Z}}^\star = \vec{\mathbf{X}} - \frac{1}{\mu}(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star$, we have

$$\begin{aligned}
\frac{\mathrm{d}\vec{\mathbf{Z}}^\star}{\mathrm{d}(\cdot)} &= \frac{\mathrm{d}(\vec{\mathbf{X}} - \frac{1}{\mu}(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star)}{\mathrm{d}(\cdot)} \\
&= \frac{\partial(\vec{\mathbf{X}} - \frac{1}{\mu}(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star)}{\partial(\cdot)} + \frac{\partial(\vec{\mathbf{X}} - \frac{1}{\mu}(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star)}{\partial\vec{\mathbf{Z}}^\star}\frac{\mathrm{d}\vec{\mathbf{Z}}^\star}{\mathrm{d}(\cdot)} \\
&= \frac{\partial(\vec{\mathbf{X}} - \frac{1}{\mu}(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star)}{\partial(\cdot)} - \frac{1}{\mu}\frac{\partial(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star}{\partial\vec{\mathbf{Z}}^\star}\frac{\mathrm{d}\vec{\mathbf{Z}}^\star}{\mathrm{d}(\cdot)}.
\end{aligned}$$

Rearrange the above equation, we have

$$\left(\mathbf{I} + \frac{1}{\mu}\frac{\partial(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star}{\partial\vec{\mathbf{Z}}^\star}\right)\frac{\mathrm{d}\vec{\mathbf{Z}}^\star}{\mathrm{d}(\cdot)} = \frac{\partial(\vec{\mathbf{X}} - \frac{1}{\mu}(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star)}{\partial(\cdot)},$$

which implies that

$$\begin{aligned}
\frac{\mathrm{d}\vec{\mathbf{Z}}^\star}{\mathrm{d}(\cdot)} &= \left(\mathbf{I} + \frac{1}{\mu}\frac{\partial(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star}{\partial\vec{\mathbf{Z}}^\star}\right)^{-1}\frac{\partial(\vec{\mathbf{X}} - \frac{1}{\mu}(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star)}{\partial(\cdot)} \\
&= -\frac{1}{\mu}\left(\mathbf{I} + \frac{1}{\mu}\frac{\partial(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star}{\partial\vec{\mathbf{Z}}^\star}\right)^{-1}\frac{\partial(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star}{\partial(\cdot)}.
\end{aligned}$$

Let

$$J|_{\vec{\mathbf{Z}}^\star} = \mu\left(\mathbf{I} + \frac{1}{\mu}\frac{\partial(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star}{\partial\vec{\mathbf{Z}}^\star}\right) = \left(\mu\mathbf{I} + \frac{\partial(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star}{\partial\vec{\mathbf{Z}}^\star}\right),$$

we obtain

$$\begin{aligned}
\frac{\partial\ell}{\partial(\cdot)} &= \frac{\partial\ell}{\partial\vec{\mathbf{Z}}^\star}\frac{\mathrm{d}\vec{\mathbf{Z}}^\star}{\mathrm{d}(\cdot)} \\
&= -\frac{\partial\ell}{\partial\vec{\mathbf{Z}}^\star}\left(J^{-1}|_{\vec{\mathbf{Z}}^\star}\right)\frac{\partial(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star}{\partial(\cdot)} \\
&= -\frac{\partial\ell}{\partial h_{\Theta^{(2)}}}\frac{\partial h_{\Theta^{(2)}}}{\partial\vec{\mathbf{Z}}^\star}\left(J^{-1}|_{\vec{\mathbf{Z}}^\star}\right)\frac{\partial(\mathbf{I}\otimes\Delta)\vec{\mathbf{Z}}^\star}{\partial(\cdot)}.
\end{aligned}$$

Note that the computation cost of the inverse Jacobian J^{-1} is expensive. To compute $-\frac{\partial \ell}{\partial \vec{\mathbf{Z}}^\star}\left(J^{-1}\big|_{\vec{\mathbf{Z}}^\star}\right)$, we can alternatively solve the linear system, which is more efficient to compute:

$$\left(J^\top\big|_{\vec{\mathbf{Z}}^\star}\right)\mathbf{u}^\top + \left(\frac{\partial \ell}{\partial \vec{\mathbf{Z}}^\star}\right)^\top = \mathbf{0}, \tag{30}$$

where the vector-Jacobian product can be efficiently obtained by automatic differentiation packages (e.g., PyTorch) for any $\mathbf{u}$ without explicitly writing out the Jacobian matrix.

D Canonical Graph Laplacian Operators and Dirichlet Energies

D.1 Canonical graph Laplcian operators

Here we discuss the connection between our definition of graph Laplacian operator given in Definition 5 with canonical graph Laplacian operators, including unnormalized graph Laplacian, random walk graph Laplacian, and normalized graph Laplacian [72, 73].

Unnormalized graph Laplacian operator. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ and a function $f : \mathcal{V} \mapsto \mathbb{R}$, the unormalized graph Laplacian operator $\Delta^{(\text{un})}$ is given by:

$$(\Delta^{(\text{un})} f)(i) = \sum_{j=1}^{N} A_{i,j} \left(f(i) - f(j) \right), \quad \text{for all } i \in \mathcal{V}. \tag{31}$$

The unnormalized graph Laplacian operator corresponds to our graph Laplacian operator when we adopt the following fixed positive real-value functions χ, ϕ and φ:

$$\begin{aligned} \chi(i) &= 1, \quad \text{for all } i \in \mathcal{V}, \\ \varphi^2([i,j])\phi([i,j]) &= A_{i,j}, \quad \text{for all } [i,j] \in \mathcal{E}, \end{aligned} \tag{32}$$

which can be easily verified by substituting the above χ, ϕ, φ into Eq. (7) to obtain Eq. (31).

Random walk graph Laplacian operator. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ and a function $f : \mathcal{V} \mapsto \mathbb{R}$, the random walk graph Laplacian operator $\Delta^{(\text{r})}$ is given by:

$$(\Delta^{(\text{r})} f)(i) = \sum_{j=1}^{N} \frac{A_{i,j}}{D_i} \left(f(i) - f(j) \right), \quad \text{for all } i \in \mathcal{V}. \tag{33}$$

Similarly, the random walk graph Laplacian operator corresponds to our graph Laplacian operator when we adopt the following fixed χ, ϕ and φ:

$$\begin{aligned} \chi(i) &= D_i, \quad \text{for all } i \in \mathcal{V}, \\ \varphi^2([i,j])\phi([i,j]) &= A_{i,j}, \quad \text{for all } [i,j] \in \mathcal{E}, \end{aligned} \tag{34}$$

which can be easily verified by substituting χ, ϕ, φ in Eq. (34) into Eq. (7) to obtain Eq. (33).

Normalized graph Laplacian operator [72, 33]. Following the same inner product on vertex space and edge space defined in Definitions 1 and 2 respectively, the form of the graph gradient operator used in the normalized graph Laplacian operator is different from our graph gradient operator defined in Definition 3.

Definition 6 (Normalized Graph Gradient)**.** *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a function $f : \mathcal{V} \mapsto \mathbb{R}$, the normalized graph gradient $\nabla^{(n)} : \mathcal{H}(\mathcal{V}, \chi) \mapsto \mathcal{H}(\mathcal{E}, \phi)$ is defined as follows:*

$$(\nabla^{(n)} f)([i,j]) = \varphi([i,j]) \left(\frac{f(j)}{\sqrt{D_j}} - \frac{f(i)}{\sqrt{D_i}} \right), \textit{ for all } [i,j] \in \mathcal{E}, \tag{35}$$

The normalized graph divergence operator $\text{div}^{(\text{n})} : \mathcal{H}(\mathcal{E}, \phi) \mapsto \mathcal{H}(\mathcal{V}, \chi)$ can be derived by satisfying $\langle \nabla^{(\text{n})} f, g \rangle_{\mathcal{E}} = \langle f, -\text{div}^{(\text{n})} g \rangle_{\mathcal{V}}$ for any functions $f : \mathcal{V} \mapsto \mathbb{R}$ ad $g : \mathcal{E} \mapsto \mathbb{R}$. Based on the normalized graph gradient operator and normalized graph divergence operator, the normalized graph Laplacian operator $\Delta^{(\text{n})}$ is given by [10]:

$$(\Delta^{(\text{n})} f)(i) = \sum_{j=1}^{N} \frac{A_{i,j}}{\sqrt{D_i}} \left(\frac{f(i)}{\sqrt{D_i}} - \frac{f(j)}{\sqrt{D_j}} \right), \quad \text{for all } i \in \mathcal{V}. \tag{36}$$

Then, the normalized graph Laplacian operator corresponds to the case when we adopt the following fixed χ, ϕ and φ:

$$\begin{aligned} \chi(i) &= \sqrt{D_i} \quad \text{for all } i \in \mathcal{V}, \\ \varphi^2([i,j])\phi([i,j]) &= A_{i,j}, \quad \text{for all } [i,j] \in \mathcal{E}. \end{aligned} \tag{37}$$

We omit the detail derivation here, which should be easy to verify following a similar way to derive the explicit expression of the graph Laplacian operator given in Lemma 2.

Remark 5 (The Hilbert spaces for canonical graph Laplacian operators)**.** *It is worth noting that choosing the above canonical graph Laplacian operators $\Delta^{(un)}, \Delta^{(rw)}$, or $\Delta^{(n)}$, the functions ϕ and φ are not fixed. Therefore, the corresponding Hilbert space on edges and the graph gradient operator are not explicitly specified, which could lead to confusion in modeling the graph geometry. As a result, graph learning methods that are developed based these canonical Laplacian operators, for examples GNNs [2, 8, 22], graph-based semi-supervised learning algorithms [33, 62, 61], may have limited adaptability to learn graph geometry and potentially hinder their performance in graph learning problems. On the contrary, in the definition of our graph Laplacian operator, we use χ, ϕ, φ to explicitly learn the Hilbert space on vertices, the Hilbert space on edges, and the graph gradient operator, respectively. Therefore, it may have more flexibility to model the graph geometry.*

D.2 Canonical Dirichlet energies

Here we introduce the Dirichlet energies induced by canonical graph Laplacian operators.

Dirichlet energy induced by unnormalized and random walk graph Laplacians operators. For the unnormalized and random walk graph Laplacian operators $\Delta^{(\mathrm{un})}, \Delta^{(\mathrm{rw})}$, $\varphi^2([i,j])\phi([i,j]) = A_{i,j}$ for all $[i,j] \in \mathcal{E}$. Then by Eq. (13), the Dirichlet energy $\mathcal{S}^{(\mathrm{un})}$ (or $\mathcal{S}^{(\mathrm{rw})}$) induced by $\Delta^{(\mathrm{un})}$ (or $\Delta^{(\mathrm{rw})}$) is given by

$$\begin{aligned}\mathcal{S}^{(\mathrm{un})}(f) = \mathcal{S}^{(\mathrm{r})}(f) &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\varphi([i,j])^2\phi([i,j])\|f(j)-f(i)\|^2 \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}A_{i,j}\|f(j)-f(i)\|^2. \end{aligned} \tag{38}$$

By Lemma 9, we have

$$\begin{aligned}\left.\frac{\mathrm{d}\mathcal{S}^{(\mathrm{un})}(f)}{\mathrm{d}f}\right|_i = \left.\frac{\mathrm{d}\mathcal{S}^{(\mathrm{r})}(f)}{\mathrm{d}f}\right|_i &= 2\sum_{j=1}^{N}\varphi([i,j])^2\phi([i,j])(f(i)-f(j)) \\ &= 2\sum_{j=1}^{N}A_{i,j}(f(i)-f(j)) \\ &= 2\Delta^{(\mathrm{un})}f(i) = 2D_i\Delta^{(\mathrm{r})}f(i). \end{aligned} \tag{39}$$

Dirichlet energy induced by the normalized graph Laplacian operator. For the normalized graph Laplacian operator $\Delta^{(\mathrm{n})}$, $\varphi^2([i,j])\phi([i,j]) = A_{i,j}$ for all $[i,j] \in \mathcal{E}$. The Dirichlet energy $\mathcal{S}^{(\mathrm{n})}$ induced by $\Delta^{(\mathrm{n})}$ is given by

$$\begin{aligned}\mathcal{S}^{(\mathrm{n})}(f) &= \|\nabla^{(\mathrm{n})}f\|_{\mathcal{E}}^2 \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\varphi([i,j])^2\phi([i,j])\left\|\frac{f(j)}{\sqrt{D_j}}-\frac{f(i)}{\sqrt{D_i}}\right\|^2 \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}A_{i,j}\left\|\frac{f(j)}{\sqrt{D_j}}-\frac{f(i)}{\sqrt{D_i}}\right\|^2. \end{aligned} \tag{40}$$

Then we have

$$\begin{aligned}\left.\frac{\mathrm{d}\mathcal{S}^{(\mathrm{n})}(f)}{\mathrm{d}f}\right|_i &= 2\sum_{j=1}^{N}\frac{A_{i,j}}{\sqrt{D_i}}\left(\frac{f(i)}{\sqrt{D_i}}-\frac{f(j)}{\sqrt{D_j}}\right) \\ &= 2\Delta^{(\mathrm{n})}f(i). \end{aligned} \tag{41}$$

E Experimental Setup and Additional Experiments

E.1 Experimental setup

E.1.1 Dataset description

Semi-supervised node classification. For semi-supervised node classification, we use both homophilic and heterophilic datasets to demonstrate the effectiveness of our framework. The homophilic datasets are the citation networks including Cora, CiteSeer, and PubMed. Regarding the heterophilic datasets, we avoid using very small datasets due to the potential sensitivity of GNN performance to different random splitting settings and the risk of overfitting. Therefore, we utilize larger heterophilic datasets such as Chameleon, Squirrel, Penn94, Cornell5, and Amherst41, each containing at least 1000 nodes in the graph. More information on these new heterophilic datasets can be found in [40]. Dataset statistics are provided in the Caption of Tables 2 and 5.

Graph classification. For the graph classification task, we utilize well-known datasets, including MUTAG, PTC, PROTEINS, NCI1, IMDB-BINARY, and IMDB-MULTI, which are available in the TUDataset benchmark suite [74]. MUTAG, PTC, PROTEINS, and NCI1 are collected from the bioinformatics domain while IMDB-BINARY and IMDB-MULTI are social network datasets. Specifically, graphs in MUTAG represent mutagenic aromatic and heteroaromatic compounds. PTC consists of compounds with carcinogenicity for males and females. NCI1 dataset consists of chemical compounds with the ability to suppress the growth of tumor cells made available by the National Cancer Institute (NCI). PROTEINS dataset consists of graphs with nodes as secondary structure elements (SSEs) with adjacent nodes indicating the neighbourhood of the amino-acid sequence or 3D space. For all the graph classification datasets, we follow the 10-fold splits from [24] for training, validation, and testing.

E.1.2 Model architectures

For graph classification tasks and node classification tasks on homophilic graphs, our model architecture consists of a 1-layer MLP layer and an implicit graph diffusion layer followed by an output classifier. For node classification tasks on heterophilic graphs, our model architecture consists of an input feature preprocessing unit layer and an implicit graph diffusion layer followed by an output classifier. Specifically,, we use $h_{\Theta^{(1)}}(\mathbf{A}, \mathbf{X}) = \mathbf{AX}$ as preprocessing step for Chameleon and Squirrel datasets. For Penn94, Cornell91 and Amherst41 datasets, we use the LINKX structure of $\text{MLP}(\text{Concat}(\text{MLP}(\mathbf{A}), \text{MLP}(\mathbf{X})))$ as a preprocessing step. For a fair comparison, we add the same LINKX preprocessing step to other Implicit GNN models and report the best scores achieved, with and without the preprocessing step. We use dropout after each layer and batch normalization before the Implicit layer. We implement our code in Pytorch [75] along with PyTorch-geometric libraries [38]. For heterophilic datasets of Penn94, Cornell41, and Amherst41, we use the source codes of other baselines and report the best results obtained after a substantial hyperparameter search.

E.1.3 Hyperparameters

We search the hyperparameters for our model from the following.

- hidden dimension $\in \{16, 32, 64, 128\}$
- $\mu \in \{1.25, 2.1, 2.2, 2.4, 2.5, 5, 7.5\}$.
- learning rate $\in \{0.001, 0.005, 0.01\}$
- dropout rate $\in \{0, 0.1, 0.25, 0.5, 0.75\}$
- maximum iteration of the implicit layer $\in \{2, 4, 10, 20, 100\}$
- weight decay $\in \{0, 1e\text{-}4, 1e\text{-}5\}$

The optimal hyperparameters are shown in Tables 6 and 7.

E.2 Results for the learned graph geometry

In this subsection, we compare the graph geometry induced by the learned positive real-valued functions and the graph geometry induced by the original edge weights for several heterophilic

Table 6: Hyperparameters of DIGNN-$\Delta^{(\mathrm{rw})}$.

Dataset	μ	num hid	lr	weight decay	max iter	threshold	dropout
Chameleon	2.20	128	0.01	0	10	1e-6	0.5
Squirrel	2.20	128	0.01	0	10	1e-6	0.1
Penne94	2.10	32	0.001	0	10	1e-6	0.75
Cornell5	2.50	32	0.001	0	4	1e-6	0.75
Amherst41	2.50	64	0.01	1e-4	20	1e-6	0.5
Cora	2.10	64	0.01	1e-5	20	1e-6	0.75
CiteSeer	2.10	64	0.001	0	10	1e-6	0.5
PubMed	2.50	64	0.001	1e-5	10	1e-6	0.5
MUTAG	2.40	64	0.005	1e-5	20	1e-6	0
PTC	2.50	64	0.001	0	10	1e-6	0
Proteins	2.50	64	0.001	0	10	1e-6	0
NCI1	2.40	64	0.001	1e-5	4	1e-6	0
IMDB-B	2.50	128	0.001	0	10	1e-6	0
IMDB-M	2.50	64	0.001	1e-5	20	1e-6	0
PPI	2.00	256	0.01	0	50	1e-6	0.1

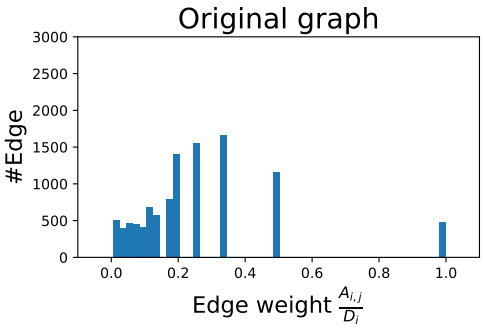
Table 7: Hyperparameters of DIGNN-Δ_{Φ}.

Dataset	μ	num hid	lr	weight decay	max iter	threshold	dropout
Chameleon	2.20	128	0.01	0	10	1e-6	0.5
Squirrel	2.20	128	0.01	0	10	1e-6	0.1
Penne94	1.25	32	0.001	0	10	1e-6	0.75
Cornell5	2.50	32	0.001	0	4	1e-6	0.75
Amherst41	2.50	64	0.01	0	20	1e-6	0.5
Cora	2.10	64	0.01	1e-5	20	1e-6	0.75
CiteSeer	2.10	64	0.001	0	10	1e-6	0.5
PubMed	2.50	64	0.001	1e-5	10	1e-6	0.5
MUTAG	1.25	64	0.005	1e-5	20	1e-6	0
PTC	1.25	64	0.001	0	10	1e-6	0
Proteins	1.25	64	0.001	0	10	1e-6	0
NCI1	1.25	64	0.001	1e-5	4	1e-6	0
IMDB-B	2.50	128	0.001	0	10	1e-6	0
IMDB-M	1.25	64	0.001	1e-5	20	1e-6	0
PPI	2.00	256	0.01	0	50	1e-6	0.1

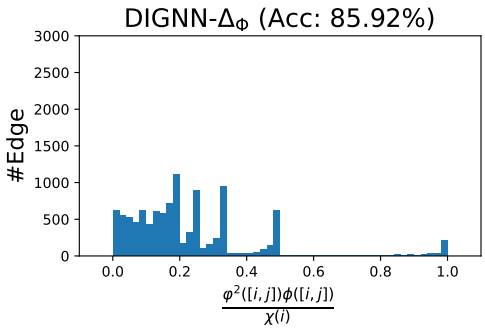
and homophilic benchmark datasets. We present the distributions of the values of $\frac{\varphi^2([i,j])\phi([i,j])}{\chi(i)}$ of DIGNN-Δ_{Φ} over all edges and the distribution of original edge weights in Figure 2.

We observe from Figures 2a to 2d that the distributions of the values of $\frac{\varphi^2([i,j])\phi([i,j])}{\chi(i)}$ on homophilic benchmark datasets, i.e., Cora and CiteSeer, are similar to the distributions of original normalized edge weights, which indicates that aggregating and transforming node features over the original graph topology is highly beneficial for label prediction. It explains why our models and explicit GNN baselines archive similar performance on these homophilic benchmark datasets.

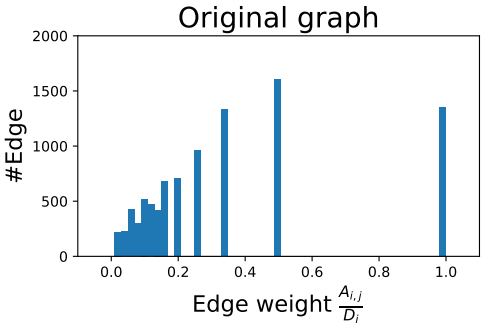
Contradict to the results of Figures 2e to 2h on homophilic graphs, the results on heterophilic graphs, i.e., Chameleon and Squirrel, show that the distributions of the values of $\frac{\varphi^2([i,j])\phi([i,j])}{\chi(i)}$ are substantially different from the distributions of the original edge weights. We observe that the values learned by DIGNN-Δ_{Φ} are mostly close to zero for most edges, indicating that the propagation weights of DIGNN-Δ_{Φ} are generally very small. It illustrates that the local neighborhood information in Chameleon and Squirrel is not as helpful for label prediction as it is in Cora and CiteSeer.



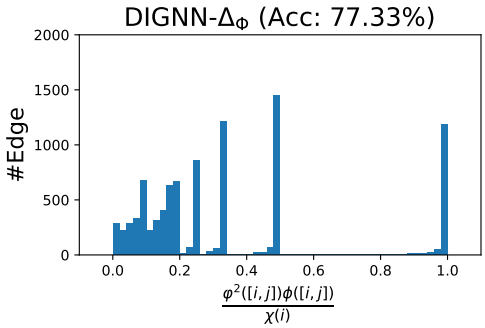
(a) Original edge weight of Cora.



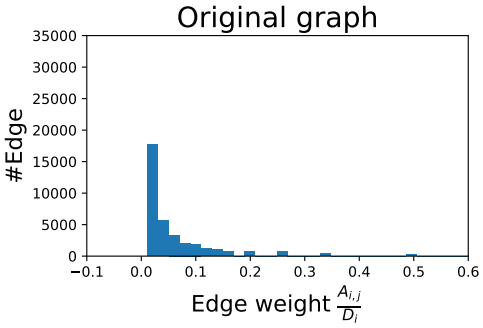
(b) Learned χ, ϕ, φ for Cora.



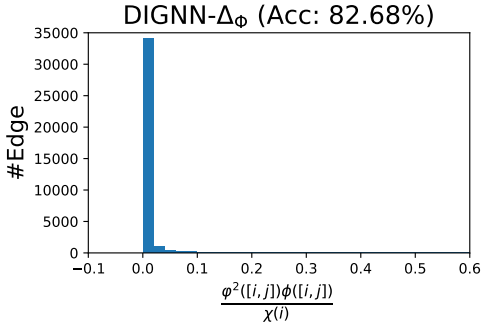
(c) Original edge weight of CiteSeer.



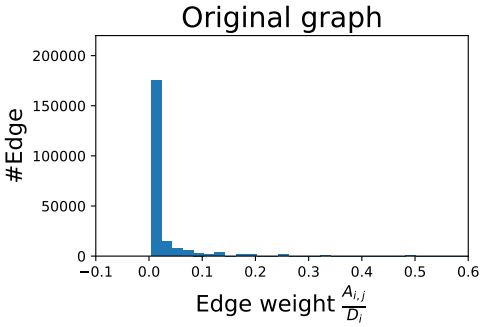
(d) Learned χ, ϕ, φ for CiteSeer.



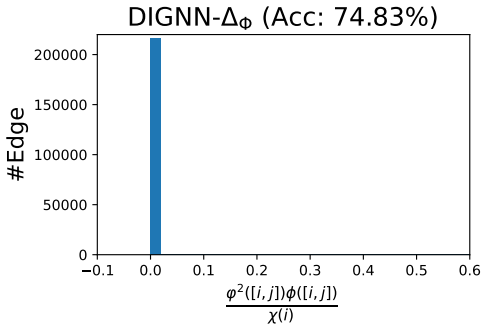
(e) Original edge weight of Chameleon.



(f) Learned χ, ϕ, φ for Chameleon.



(g) Original edge weight of Squirrel.



(h) Learned χ, ϕ, φ for Squirrel.

Figure 2: Results.

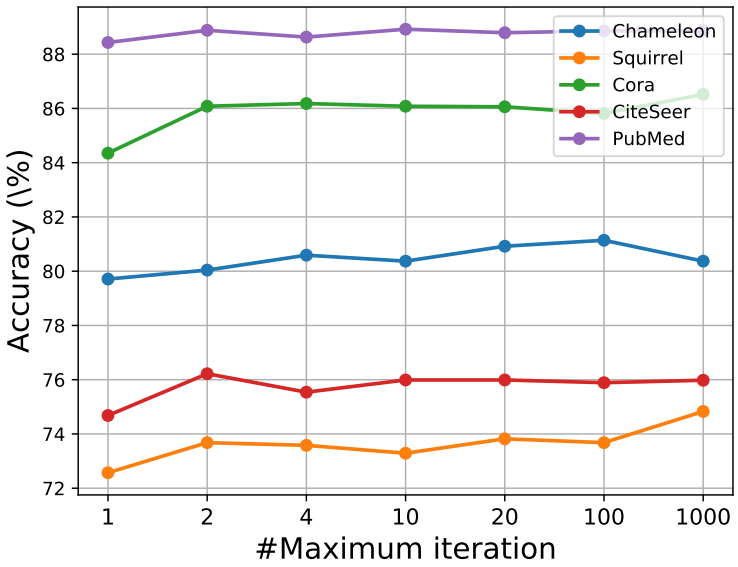


Figure 3: Node classification performance of DIGNN-Δ_Φ with different maximum iteration numbers.

Consequently, propagating and transforming node features over the original graph topology may hinder the performance of GNNs on these heterophilic datasets.

E.3 Results with different iteration numbers

We conduct experiments on several heterophilic and homophilic datasets, including Chameleon, Squirrel, Cora, CiteSeer, and PubMed, to study the performance of DIGNN-Δ_Φ with different maximum iteration numbers. The results are presented in Figure 3. Figure 3 shows that the performance of DIGNN-Δ_Φ on all datasets first improves as the maximum iteration number increases from 1 to 2. However, beyond a certain point, further increasing the maximum iteration number does not lead to significant improvements and the performance stabilizes. This observation indicates that the iteration algorithm described in Equation 7 quickly converges to the equilibrium of our implicit graph diffusion layer.

E.4 Running time of DIGNNs

Table 8 reports the averaged running time of DIGNN-$\Delta^{(\mathrm{rw})}$ and DIGNN-Δ_Φ on several homophilic and heterophilic benchmark datasets. Our experiments are conducted on a server equipped with a GeForce RTX 2080 Ti and 12GB of GPU memory.

Table 8: Running of DIGNNs on node classification datasets. Averaged running time per epoch (ms) / averaged total running time (s).

Method	Chameleon	Squirrel	Cora	CiteSeer	PubMed
DIGNN-$\Delta^{(\mathrm{rw})}$	22.38ms / 32.78s	60.58ms / 104.25s	22.02ms / 30.35s	23.79ms / 36.08s	34.82ms / 51.21s
DIGNN-Δ_Φ	38.89ms / 67.74s	133.46ms / 249.48s	28.92ms / 46.38s	30.95ms / 40.62s	71.11ms / 123.61s