Optimization-induced Implicit Graph Diffusion

Anonymous Authors[1]

Abstract

Due to the over-smoothing issue, most existing graph neural networks (GNNs) can only capture limited dependencies with their inherently finite aggregation layers. To overcome this limitation, we propose a new kind of graph convolution, called Optimization-induced Implicit Graph Diffusion (OIGD), which implicitly has access to infinite hops of neighbors while adaptively aggregating features with nonlinear diffusion to prevent over-smoothing. Notably, we show that the learned representation can be formalized as the minimizer of an explicit convex optimization objective. With this property, we can theoretically characterize the equilibrium of our OIGD from an optimization perspective. More interestingly, we can induce new structural variants by modifying the corresponding optimization objective. To be specific, we can embed prior properties to the equilibrium, as well as introducing skip connections to promote training stability. Extensive experiments show that OIGD is good at capturing long-range dependencies, and performs well on both homophilic and heterophilic graphs with nonlinear diffusion. Moreover, we show that the optimization-induced variants of our models can boost the performance and improve training stability and efficiency as well. As a result, our OIGD obtains significant improvements on both node-level and graph-level tasks.

1. Introduction

In recent years, Graph Neural Networks (GNNs) rise to be the state-of-the-art models for graph mining (Kipf & Welling, 2016; Veličković et al., 2017; Hamilton et al., 2017) and have extended applications in various scenarios such as biochemical structure discovery (Gilmer et al., 2017; Wan et al., 2019), recommender systems (Ying et al., 2018; Fan et al., 2019), natural language processing (Gao et al., 2019; Zhu et al., 2019), and computer vision (Pang & Cheung, 2017; Valsesia et al., 2020). Despite the success, these GNNs typically lack the ability to capture long-range dependencies. In particular, the common message-passing GNNs can only aggregate information from T-hop neighbors with T propagation steps. However, existing works have observed that GNNs often degrade catastrophically when propagation steps $T \geq 2$ (Li et al., 2018), a phenomenon widely characterized as over-smoothing. Several works try to alleviate it with more expressive aggregations of higher-order neighbors (Abu-El-Haija et al., 2019; Zhu et al., 2020). Nevertheless, their ability to capture global information is inherently limited by the finite propagation steps.

Recently, implicit GNNs provide a new solution to this problem by replacing the deeply stacked explicit propagation layers with an implicit layer, which is equivalent to *infinite* propagation steps (Gu et al., 2020; Liu et al., 2021; Park et al., 2021). Thereby, implicit GNNs could capture very long range dependency and benefit from the global receptive field. Specifically, implicit GNNs achieve this by regularizing the explicit forward propagation to a root-finding problem with convergent equilibrium states. They further adopt implicit differentiation (Krantz & Parks, 2012) directly through the equilibrium states to avoid long range back-propagation. As a result, the methods could get rid of performance degradation caused by explosive variance with more depth (Zhou et al., 2021) while having constant memory footprint even with infinite propagation steps. These advantages indicate that implicit GNNs are promising alternatives to existing explicit GNNs.

The performance of implicit GNNs is largely determined by the design of the implicit layer. However, it is overlooked in previous works. Their implicit layers are direct adaptations of recurrent GNNs (Gu et al., 2020) and lack theoretical justifications of their diffusion properties. Notably, a major drawback is that their aggregation mechanisms correspond to a *linear isotropic diffusion* that treats all neighbors equally. However, as noted by recent theoretical discussions, this isotropic property is exactly the cause of the over-smoothing issue (Oono & Suzuki, 2019). Thus, it is still hard for them to benefit from long range dependency. This problem reveals that the infinite depth itself is inadequate. The design of the diffusion process, which decides the quality of the equilibrium states, is also the key to the actual performance of implicit GNNs.

Motivated by this situation, in this work, we propose a novel and principled implicit graph diffusion layer, whose advantages are two folds. **First**, drawing inspirations from anisotropic diffusion process like PM diffusion (Perona & Malik, 1990), we extend the linear isotropic diffusion to

a more expressive *nonlinear diffusion* mechanism, which learns nonlinear flux features between node pairs before aggregation. In particular, our design of the nonlinear diffusion ensures that more information can be aggregated from similar neighbors and less from dissimilar neighbors, making node features less likely to over-smooth. **Second**, we can show for the first time that the equilibrium of our implicit nonlinear diffusion is the solution to a convex optimization objective. Based on this perspective, we can not only characterize theoretical properties of the equilibrium states, but also derive new structural variants in a principled way, *e.g.*, adding regularization terms to the convex objective.

Based on the above analysis, we propose a model named Optimization-induced Implicit Graph Diffusion (OIGD). Several recent works have tried to connect GNN propagations to structural optimization objectives (Zhu et al., 2021; Yang et al., 2021). However, their frameworks only consider linear isotropic diffusion steps and ignore the nonlinear transformation of features, which, however, is also crucial in GNNs. In comparison, our OIGD admits a unified objective of both the *nonlinear* diffusion step and the transformation step. Therefore, our framework is more general, as it could take the interaction between diffusion and transformation into consideration. Last but not least, compared to previous optimization-based GNNs whose propagation rule is inspired by one single optimization step, our OIGD directly models the equilibrium of the implicit layer as a minimizer of a convex objective (thus we call it optimization-induced). This shows that our OIGD enjoys a much closer connection to the optimization objective compared to previous works.

We evaluate OIGD on a wide range of benchmark datasets including both node-level and graph-level classification tasks. The results demonstrate that our OGIN effectively captures long-range dependency and outperforms both explicit and implicit GNN models in most cases. In particular, on heterophilic graphs, our nonlinear diffusion achieves significant improvements over previous implicit GNNs with isotropic diffusion. And we further verify that two structural variants of OIGD induced by principled feature regularization can indeed obtain clear improvements over the vanilla OIGD, which demonstrates the usefulness of our optimization framework. In summary, our contributions are:

- We develop a new kind of implicit GNNs, OIGD, whose nonlinear diffusion overcomes the limitations of existing linear isotropic diffusion by adaptively aggregating nonlinear features from similar neighbors.
- We develop the first optimization framework for an implicit GNN by showing that the equilibrium states of OIGD correspond to the solution to a convex objective. Based on this perspective, we derive three principled structural variants with empirical benefits.
- Extensive experiments on node-level and graph-level classification tasks show our OIGD obtains state-of-the-art performance among implicit GNNs, and also compares favorably to explicit GNNs.

2. Preliminary on Implicit GNNs

Consider a general graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V}$ and edge set $\mathcal{E}$, where $|\mathcal{V}| = n$ and $|\mathcal{E}| = m$. Denote node features as $\mathbf{X} \in \mathbb{R}^{n\times p}$, and the adjacency matrix as $\mathbf{A} \in \mathbb{R}^{n\times n}$, where $\mathbf{A}_{i,j} = 1$ if $(i,j) \in \mathcal{E}$ and $\mathbf{A}_{i,j} = 0$ otherwise. Denote the normalized adjacency matrix as $\hat{\tilde{\mathbf{A}}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$. Here $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the augmented adjacency matrix, and $\tilde{\mathbf{D}} = \mathrm{diag}(d_1, \ldots, d_n)$ is the degree matrix of $\tilde{\mathbf{A}}$, where $d_j = \sum_j \tilde{\mathbf{A}}_{i,j}$.

Recently, several works (Gu et al., 2020; Liu et al., 2021; Park et al., 2021) have studied implicit GNNs. They are motivated by recurrent GCNs that employ the same transformation in each layer as follows:

$$\mathbf{Z}_{k+1} = \sigma(\hat{\tilde{\mathbf{A}}}\mathbf{Z}_k\mathbf{W} + b_{\mathbf{\Omega}}(\mathbf{X})), \quad k = 1, 2, \ldots, \infty, \quad (1)$$

where $\mathbf{W}$ is a weight matrix, $b_{\mathbf{\Omega}}(\mathbf{X})$ is the embedding of the input features $\mathbf{X}$ through an affine transformation parameterized by $\mathbf{\Omega}$, and $\sigma(\cdot)$ is an element-wise activation function. In fact, it models the limiting process when the above recurrent iteration is applied for infinite times, *i.e.*, $\mathbf{Z} = \lim_{k\to\infty} \mathbf{Z}_k$. As a result, the final equilibrium $\mathbf{Z}$ potentially contains global information from all neighbors in the graph. Different from explicit GNNs, the output features $\mathbf{Z}$ of a general implicit GNN are directly modeled as the solution of the following equation,

$$\mathbf{Z} = \sigma(\hat{\tilde{\mathbf{A}}}\mathbf{Z}\mathbf{W} + b_{\mathbf{\Omega}}(\mathbf{X})). \quad (2)$$

The prediction of the implicit models is given by $\mathbf{Y} = g_{\mathbf{\Theta}}(\mathbf{Z})$, where $g_{\mathbf{\Theta}}$ is some trainable function parameterized by $\mathbf{\Theta}$.

In practice, in the forward pass, we can directly find the equilibrium $\mathbf{Z}$ via off-the-shelf black-box solvers like Broyden's method (Broyden, 1965). While in the backward pass, one can also analytically differentiate through the equilibrium by the implicit function theorem (Krantz & Parks, 2012), which does not require storing any intermediate activation values and uses only constant memory (Bai et al., 2019). Several alternative strategies have also been proposed to improve its training stability (Geng et al., 2021).

Limitations. As shown above, existing implicit GNN models adopt the same feature propagation as the canonical GCN (Kipf & Welling, 2016). As discussed in many previous works (Oono & Suzuki, 2019), this linear isotropic propagation matrix $\hat{\tilde{\mathbf{A}}}$ will inevitably lead to feature oversmoothing after infinite propagation steps. Although this problem could be partly addressed by the implicit propagation process that admits a meaningful fixed point $\mathbf{Z}^*$, the isotropic nature will still degrade the propagation process by

introducing extra feature noises from dissimilar neighbors, as we elaborate below.

3. Proposed Implicit Graph Diffusion

In view of the limitations of existing implicit GNNs based on linear isotropic diffusion, in this section, we propose a new nonlinear graph diffusion process for the design of implicit GNNs. Inspired by anisotropic diffusion equation, our nonlinear diffusion could adaptively aggregate more features from similar neighbors while separating dissimilar neighbors. Thus, its equilibrium states will preserve more discriminative features.

3.1. Nonlinear Diffusion Equation

Formulation. Given a general continuous manifold $\mathcal{M}$ where a feature function $\mathbf{u}$ resides, along with operators such as the gradient and divergence that reside on the manifold $\mathcal{M}$, one can model diffusion processes on $\mathcal{M}$ (Eliasof et al., 2021; Chamberlain et al., 2021). In particular, we consider a *nonlinear diffusion* at time t:

$$\partial_t \mathbf{u} = \operatorname{div}(\mathcal{K}^* \sigma(\mathcal{K} \nabla \mathbf{u})), \tag{3}$$

where $\mathcal{K}$ is a linear operator, $\mathcal{K}^*$ is its adjoint operator, ∇ is the gradient operator, and $\sigma(\cdot)$ is an element-wise transformation. It can be understood as a composition of two sequential operations, one is the nonlinear flux (the direction of the flow) $\mathbf{j}$ induced by the gradient $\nabla \mathbf{u}$, *i.e.*,

$$\mathbf{j} = -\mathcal{K}^* \sigma(\mathcal{K} \nabla \mathbf{u}), \tag{4}$$

and the other is the continuity equation satisfying

$$\partial_t \mathbf{u} = -\operatorname{div} \mathbf{j}. \tag{5}$$

Generalization of Isotropic Diffusion. One notable degenerated case is the linear isotropic diffusion, where we choose σ and $\mathcal{K}$ to be identity mappings in our nonlinear diffusion,

$$\partial_t \mathbf{u} = \operatorname{div}(\nabla \mathbf{u}) = \Delta \mathbf{u}. \tag{6}$$

It admits a closed-form solution $\mathbf{u}(t) = e^{-t\Delta}\mathbf{u}_0$ with an initial value $\mathbf{u}_0$ at $t = 0$, where Δ is the Laplacian operator. Wang et al. (2021) recently show that GCN propagation is equivalent to a time-discretized version of this equation. From this perspective, our nonlinear diffusion is a nontrivial generalization of isotropic diffusion in two ways: first, we add a linear operator $\mathcal{K}$ inside the Laplacian operator for flexible parameterization (Nitzberg & Shiota, 1992), and second, we introduce a nonlinear activation function σ to model nonlinear flux, which greatly enhances its expressiveness as in neural networks.

Anisotropic Property. Notably, this generalization enables us to incorporate anisotropic properties into the graph diffusion through specific choice of the activation function $\sigma(\cdot)$. To see this, ignoring $\mathcal{K}$, we have $\mathbf{j} = \sigma(\nabla \mathbf{u})$ and the following relationship on their L^2 norms

$$\|\mathbf{j}\|^2 = \|\sigma(\nabla \mathbf{u})\|^2 = \left(\int \left[\sigma(\nabla \mathbf{u}(x))\right]^2 dx \right)^{1/2} \tag{7}$$

In this case, comparing $\sigma(x) = x$ and $\sigma(x) = \tanh(x)$ as in Figure 1, we can see that the nonlinear activation $\tanh$ will keep more small-value gradients while shrinking large-value gradients to bounded values within $[0, 1]$. In graph diffusion, this amounts to a desirable anisotropic-like behavior that adaptively aggregates more from similar neighbors and less from dissimilar neighbors, which helps prevent over-smoothing and noisy inputs. Besides, compared with PM diffusion (Perona & Malik, 1990) that manually chooses a function for the diffusion coefficient, our nonlinear diffusion with the linear operator $\mathcal{K}$ allows us to parameterize a flexible and learnable anisotropic nonlinear aggregation function.

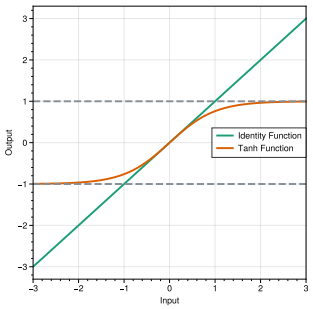

Figure 1. Comparison of two activation functions: $\sigma(x) = x$ and $\sigma(x) = \tanh(x)$.

3.2. Proposed Implicit GNN

The discussion above shows that our nonlinear diffusion is a principled generalization of previous linear isotropic diffusion with beneficial anisotropic properties. In this part, we apply it on the graph data and develop the corresponding implicit graph neural networks.

Nonlinear Diffusion on Graphs. As known in previous literature (Chung & Graham, 1997), the differential operators could be instantiated on a discrete graph $\mathcal{G}$ with n nodes and m edges, and each node contains h-dimensional features. Specifically, let $\mathbf{U} \in \mathbb{R}^{n \times h}$ be the feature matrix, the gradient operator ∇ corresponds to the incidence matrix $\mathbf{G} \in \mathbb{R}^{m \times n}$. It is defined as $\mathbf{G}_{k,i} = 1$ if edge k enters node i, -1 if it leaves node i, and 0 otherwise. The divergence operator, which is the adjoint of the gradient operator, now corresponds to $\mathbf{G}^\top \in \mathbb{R}^{n \times m}$. The linear operator $\mathcal{K}$ corresponds to a feature transformation matrix $\mathbf{K} \in \mathbb{R}^{h \times h}$, and its adjoint $\mathcal{K}^\star$ becomes $\mathbf{K}^\top$. As a result, our nonlinear diffusion in Equation (3) has the following matrix form on graph $\mathcal{G}$,

$$\partial_t \mathbf{U} = \mathbf{G}^\top \sigma(\mathbf{G}\mathbf{U}\mathbf{K}^\top)\mathbf{K}. \tag{8}$$

Following the analysis above, we choose the activation function $\sigma(x) = \tanh(x)$. A more detailed derivation can be found in Appendix B.

Our Implicit GNNs. By adopting the nonlinear graph diffusion developed above for the implicit graph diffusion mechanism, we develop a new implicit GNN, **O**ptimization-induced **I**mplicit **G**raph **D**iffusion (OIGD), with the following formulation,

$$\mathbf{Z} = \hat{\mathbf{G}}^\top \sigma(\hat{\mathbf{G}}(\mathbf{Z} + b_{\mathbf{\Omega}}(\mathbf{X}))\mathbf{K}^\top)\mathbf{K}, \tag{9a}$$

$$\hat{\mathbf{Y}} = g_{\mathbf{\Theta}}(\mathbf{X} + \mathbf{Z}), \tag{9b}$$

where $\hat{\mathbf{G}} = \mathbf{G}\tilde{\mathbf{D}}^{-1/2}/\sqrt{2}$ is normalized incidence matrix. Here, we first embed the input feature matrix $\mathbf{X}$ with an affine transformation $b_{\mathbf{\Omega}}(\cdot)$ with parameters $\mathbf{\Omega}$. Then, the input embedding is injected to the implicit diffusion layer, whose output $\mathbf{Z}$ is the equilibrium of a nonlinear equation (Equation (9a)). Afterwards, we use $\mathbf{X} + \mathbf{Z}$, the sum of the input features and the equilibrium, to predict the labels. The readout head $g_{\mathbf{\Theta}}$ can be parametrized by a linear layer or an MLP. We also provide a row-normalization variant in Appendix D.

Notably, in OIGD, we design the equilibrium states $\mathbf{Z}$ to be the residual refinement of the input features $\mathbf{X}$ through the diffusion process, *a.k.a.* the transported mass of $\mathbf{X}$ (Weickert, 1998). As a result, starting from an initial value, the estimated transported mass $\mathbf{Z}$ could be gradually refined through the fixed point iteration of our nonlinear diffusion process. Finally, it will reach a stable equilibrium $\mathbf{Z}$ that cannot be further improved, which represents the optimal estimate of the transported mass. As a result, we can obtain better feature separability of the combined node features $\mathbf{X} + \mathbf{Z}$ after diffusion.

As an implicit model, our OIGD enjoys the benefits of general implicit GNNs, including the ability to capture long-range dependency as well as constant memory footprint. With our proposed nonlinear diffusion mechanism, it could adaptively aggregate useful features from similar neighbors and filter out noisy ones. Last but not least, as we show in the next section, the equilibrium states of our OIGD can be formalized as the solution to a convex objective, from which the name "optimization-induced" comes.

4. An Optimization Framework for Implicit GNNs

In this section, inspired by recent works on optimization-based implicit models (Xie et al., 2021), we develop the first optimization framework of implicit graph neural networks. Specifically, we show that the equilibrium states of our OIGD corresponds to the solution of a convex objective. Based on this property, we show that we can derive principled variants of OIGD through various regularization terms, which demonstrates a principled way to inject inductive bias into the model representations.

4.1. Formulation of Structural Objective

Notations. We use $\otimes$ to represent the Kronecker product, and use $\odot$ to represent element-wise product. For a matrix $\mathbf{W} \in \mathbb{R}^{p\times q}$, $\mathbf{w} = \mathrm{vec}(\mathbf{W})$ represents the vectorized form of $\mathbf{W}$ obtained by stacking its columns. We use $\|\cdot\|_2$ for the matrix operator norm and $\|\cdot\|$ for the vector ℓ_2-norm. A function $f : \mathcal{H} \to \mathbb{R} \cup \{+\infty\}$ is proper if the set $\{\mathbf{x} : f(\mathbf{x}) < +\infty\}$ is non-empty, where $\mathcal{H}$ is the Euclidean space. For a proper convex function f, its proximal operator $\mathrm{Prox}_f^\mu(\mathbf{x})$ is defined as $\{\mathbf{z} \in \mathcal{H} : \mathbf{z} = \arg\min_{\mathbf{u}} \frac{1}{2\mu}\|\mathbf{u} - \mathbf{x}\|^2 + f(\mathbf{u})\}$, and its conjugate f^* is defined as $f^*(\mathbf{y}) = \sup\{\langle \mathbf{y}, \mathbf{x}\rangle - f(\mathbf{x}) : \mathbf{x} \in \mathcal{H}\}$, where $\langle\cdot,\cdot\rangle$ is the inner product. We omit μ when $\mu = 1$ for simplicity.

For convenience, from now on, we adopt the following equivalent "vectorized" version of Equation (9a) using Kronecker product:

$$\mathbf{z} = f(\mathbf{z}) := (\mathbf{K} \otimes \hat{\mathbf{G}})^\top \sigma((\mathbf{K} \otimes \hat{\mathbf{G}})(\mathbf{z} + \mathrm{vec}(b_{\mathbf{\Omega}}(\mathbf{X})))), \tag{10}$$

where $\mathbf{z}$ denotes the vectorized equilibrium states $\mathbf{Z}$ in Equation (9a), and we use $f(\mathbf{z})$ to denote the right-hand transformation. The following theorem gives the convex objective of our implicit nonlinear diffusion.

Theorem 4.1. *Assume that the nonlinear function $\sigma(\cdot)$ is monotone and L_σ-Lipschitz,* i.e.*,*

$$0 \le \frac{\sigma(a) - \sigma(b)}{a - b} \le L_\sigma, \forall a, b \in \mathbb{R}, a \ne b, \tag{11}$$

and $\mu \ge L_\sigma \left\|\mathbf{K} \otimes \hat{\mathbf{G}}\right\|_2^2 = L_\sigma \|\mathbf{K}\|_2^2 \left\|\hat{\mathbf{G}}\right\|_2^2$. We can show that the $f(\mathbf{z})$ defined in Equation (10) is equivalent to the proximal operator of a convex objective, i.e.*, $f(\mathbf{z}) = \mathrm{Prox}_\varphi(\mathbf{z})$, where*

$$\varphi(\mathbf{z}) = \psi^*(\mathbf{z}) - \frac{1}{2}\|\mathbf{z}\|^2,$$
$$\psi(\mathbf{z}) = \mathbf{1}^\top \tilde{\sigma}((\mathbf{K} \otimes \hat{\mathbf{G}})(\mathbf{z} + \mathrm{vec}(b_{\mathbf{\Omega}}(\mathbf{X})))),$$

in which $\forall a \in \mathbb{R}, \tilde{\sigma}(a) = \int_0^a \sigma(t)dt$, applied element-wisely to vectors, and $\mathbf{1}$ is the all one vector. As a result, the solution to the equilibrium equation $\mathbf{z} = f(\mathbf{z})$ is the minimizer of the convex function $\varphi(\cdot)$.

With Theorem 4.1, we can easily establish the following sufficient condition for our model to be well-posed, i.e., the existence and uniqueness of its solution, which guarantees the convergence of iterative solvers for reaching the equilibrium.

Proposition 4.2. *Equation (9a) is guaranteed to be well-posed if $\|\mathbf{K}\|_2 \|\hat{\mathbf{G}}\|_2 < 1$. Since we already have $\|\hat{\mathbf{G}}\|_2 \le 1$, we can fulfill the condition with $\|\mathbf{K}\|_2 < 1$.*

4.2. Optimization-Inspired Variants

In previous part, we have established the structural objective of our proposed OIGD. Here, we present an intriguing

application of this perspective, that is to derive structural variants of our implicit model by injecting inductive bias in a principled way. Specifically, we study three examples: skip-connection induced by Moreau envelop, and two interpretable feature regularizations: Laplacian smoothing and feature decorrelation.

Optimization-Inspired Skip-connection. As the equilibrium is the minimizer of an objective function $\varphi(\cdot)$. Thus, we can replace it directly with a new formula $\Phi(\cdot)$, as long as it has the same minimizer as the original one. Specifically, we choose its Moreau envelope function. Given a proper closed convex function $f : \mathcal{H} \to \mathbb{R} \cup \{+\infty\}$ and $\mu > 0$, the Moreau envelope of f is the function

$$M_{\varphi}^{\mu}(\mathbf{z}) = \min_{\mathbf{u}} \frac{1}{2\mu} \|\mathbf{u} - \mathbf{z}\|^2 + \varphi(\mathbf{u}),$$

which is a smoothed approximate of φ (Attouch & Aze, 1993). Meanwhile, the Moreau envelope function keeps the same critical points as the original one. We can compute its proximal operator as follows:

$$\mathrm{Prox}_{M_{\varphi}^{\mu}}^{\lambda}(\mathbf{z}) = \mathbf{z} + \frac{\lambda}{\mu + \lambda}(\mathrm{Prox}_{\varphi}^{\mu+\lambda}(\mathbf{z}) - \mathbf{z}),$$

where $\lambda, \mu > 0$. Further letting $\lambda = \alpha$ and $\mu = 1 - \alpha$, we have the following implicit layer induced from its proximal operator:

$$\mathbf{z} = \mathcal{T}(\mathbf{z}) \coloneqq (1 - \alpha)\mathbf{z} + \alpha f(\mathbf{z}), \tag{12}$$

where $\alpha \in [0, 1)$. The operator on the right-hand side is strongly monotone and invertible, which improves stability in the iteration while keeping the equilibrium unchanged.

Optimization-Inspired Feature Regularization. Regularization has become an important part of modern machine learning algorithms. Since we know the objective, we can combine it with regularizers to introduce customized properties to the equilibrium, which is equivalent to making a new implicit composite layer by appending one layer after the original implicit layer. Formally, if we modify $\Phi(\mathbf{z})$ to $\Phi(\mathbf{z}) + \mathcal{R}(\mathbf{z})$, then the implicit layer becomes:

$$\mathbf{z} = \mathcal{T}_{\mathcal{R}} \circ \mathcal{T}(\mathbf{z}),$$

where $\mathcal{T}_{\mathcal{R}} = \mathrm{Prox}_{\mathcal{R}}$, and $\circ$ denotes the mapping composition. When the proximal is hard to calculate, we can use a gradient descent step as an approximate evaluation, which is $\mathcal{T}_{\mathcal{R}_z} \approx \mathcal{I} - \gamma \frac{\partial \mathcal{R}_z}{\partial z}$. In general, we can instantiate $\mathcal{R}_z$ as any convex function that has the preferred properties of the equilibrium. Specifically, we consider two kinds of regularization.

- **Laplacian Regularization.** The graph Laplacian operator $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is a positive semi-definite matrix (Chung & Graham, 1997). We use the (symmetric normalized) Laplacian regularization $\hat{\tilde{\mathbf{L}}} = \mathbf{I} - \hat{\tilde{\mathbf{A}}}$ to push the equilibrium of the linked nodes closer in the feature space. It is defined as follows:

$$\mathcal{R}_{\mathrm{Lap}}(\mathbf{z}) = \mathbf{z}^{\top} \tilde{\mathbf{D}}^{-\frac{1}{2}} L \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{z} = \left\| \hat{\mathbf{G}} \mathbf{z} \right\|^2.$$

- **Feature Decorrelation.** We use the feature decorrelation regularization to reduce redundant information between feature dimensions (Rodríguez et al., 2017; Ayinde et al., 2019). It is defined as follows:

$$\mathcal{R}_{\mathrm{Dec}}(\mathbf{z}) = \frac{1}{2} \left\| \hat{\mathbf{z}} \hat{\mathbf{z}}^{\top} - \mathbf{I} \right\|_F^2,$$

where $\hat{\mathbf{z}}$ is the normalized $\mathbf{z}$.

5. Efficient Training of OIGD

Training stability has been a widely existing issue for general implicit models (Bai et al., 2019; Geng et al., 2021) as well implicit GNNs (Gu et al., 2020). To train our OIGD, we adopt an efficient training strategy that works effectively in our experiments.

Forward and Backward Computation. Specifically, in the forward pass, we adopt fixed point iteration as in Gu et al. (2020); while in the forward pass, we adopt the recently developed Phantom Gradient (Geng et al., 2021) estimation, which enjoys both efficient computation and stable training dynamics.

Variance Normalization. In previous implicit models (Bai et al., 2019; 2020), LayerNorm (Ba et al., 2016b) is often applied on the output features to enhance the training stability of implicit models. Specifically, for OIGD, we apply normalization before the nonlinear activation $\sigma(\cdot)$ and drop the mean items to keep it a sign-preserving transformation:

$$\mathrm{norm}(\mathbf{v}) = \frac{\mathbf{v}}{\sqrt{\mathrm{Var}(\mathbf{v}) + \varepsilon}} \odot \boldsymbol{\gamma} \tag{13}$$

where $\boldsymbol{\gamma} > 0$ denotes positive scaling parameters and ε is a small constant. As discussed in previous works (Zhou et al., 2021), this could effectively prevent the variance inflammation issue and help stabilize the training process with increasing depth.

6. Comparison to Related Work

Here, we highlight the contributions of our proposed OIGD through a detailed comparison to related work, including explicit GNNs and implicit GNNs.

6.1. Comparison to Implicit GNNs

Diffusion Process. Prior to our work, IGNN (Gu et al., 2020) and EIGNN (Liu et al., 2021) adopt the same aggregation mechanism as GCN (Kipf & Welling, 2016), which is equivalent to an isotropic linear diffusion that is irrelevant

Type	Method	Cornell	Texas	Wisconsin	Chameleon	Squirrel
Explicit	GCN	59.19±3.51	64.05±5.28	61.17±4.71	42.34±2.77	29.0±1.10
	GAT	59.46±6.94	61.62±5.77	60.78±8.27	46.03±2.51	30.51±1.28
	JKNet	58.18±3.87	63.78±6.30	60.98±2.97	44.45±3.17	30.83±1.65
	APPNP	63.78±5.43	64.32±7.03	61.57±3.31	43.85±2.43	30.67±1.06
	Geom-GCN	60.81	67.57	64.12	60.9	38.14
	GCNII	76.75±5.95	73.51±9.95	78.82±5.74	48.59±1.88	32.20±1.06
	H2GCN	82.22±5.67	84.76±5.57	85.88±4.58	60.30±2.31	40.75±1.44
Implicit	IGNN	61.35±4.84	58.37±5.82	53.53±6.49	41.38±2.53	24.99±2.11
	EIGNN	85.13±5.57	84.60±5.41	86.86±5.54	62.92±1.59	46.37±1.39
	OIGD(ours)	**87.30±5.28**	**86.22±5.19**	**87.45±3.84**	**65.94±2.72**	**56.73±2.04**

Table 1. Results on heterophilic node classification datasets: mean accuracy (%) ± stdev over different data splits.

to neighbor features (Wang et al., 2021). CGS (Park et al., 2021) instead uses a learnable aggregation matrix to replace the original normalized adjacency matrix. However, the aggregation matrix is fixed in the implicit layer, thus the diffusion process still cannot adapt to the iteratively updating hidden states. In comparison, we design a non-linear diffusion process with anisotropic properties. As a result, it is adaptive to the processed features while updating and helps prevent over-smoothing. An additional advantage of OIGD is that we can deduce its equilibrium states from a convex objective in a principled way, while in previous works the implicit layers are usually heuristically designed.

Training. IGNN (Gu et al., 2020) adopts projected gradient descent to limit the parameters to a restricted space at each optimization step, which, still occassionally experiences diverged iterations. EIGNN (Liu et al., 2021) and CGS (Park et al., 2021) resort to linear implicit layers and reparameterization tricks to derive a closed-form solution directly, however, at the cost of degraded expressive power compared to nonlinear layers. In our OIGD, we instead enhance the model expressiveness with our proposed nonlinear diffusion. During training, we regularize the forward pass with variance normalization and adopt the Phantom Gradient (Geng et al., 2021) to obtain a fast and stable gradient estimate.

6.2. Comparison to Explicit GNNs

Diffusion-inspired GNNs. Diffusion process has also been applied to design explicit GNNs (Atwood & Towsley, 2016; Xhonneux et al., 2020; Eliasof et al., 2021; Wang et al., 2021; Chamberlain et al., 2021), where they treat GNN as a dynamical system driven by the diffusion equation and each graph convolution layer corresponds to a discretized forward step. In comparison, our OIGD replaces the explicit iterations with an implicit nonlinear diffusion layer, which admits an equilibrium that corresponds to infinite diffusion steps. As a result, we enlarge the receptive field while being free from manual tuning of the terminal time and the time interval of the diffusion equation (Wang et al., 2021).

Optimization-Inspired GNNs. Prior to our work, Zhu et al. (2021) establish a connection between different linear propagation schemes and the corresponding optimization objectives. Yang et al. (2021) also introduce a model inspired by unfolding optimization iterations of a regularized energy function. However, their discussions are limited to explicit layers with linear isotropic diffusion and ignore the (potentially nonlinear) feature transformation steps. In comparison, our OIGD discusses implicit layers, and admits a unified objective of both the nonlinear diffusion step and the transformation step. Besides, optimization-inspired explicit GNNs usually model a single iteration step, while our implicit model ensures that the obtained equilibrium is exactly the solution to the corresponding objective.

Type	Method	Cora	Citeseer	Pubmed
Explicit	GCN	85.77	73.68	88.13
	GAT	86.37	74.32	87.62
	JKNet	85.25	75.85	88.94
	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
Implicit	IGNN*	85.80	75.24	87.66
	EIGNN*	85.89	75.31	87.92
	OIGD(ours)	88.25	76.84	89.48

Table 2. Results on homophilic node classification datasets: mean accuracy (%).

7. Experiments

In this section, we conduct a comprehensive analysis on OIGD and compare it against both implicit and explicit GNN variants on various problems and datasets. We refer to Appendix E for the details of the data statistics, network architecture and training details.

7.1. Performance on Node Classification

Datasets. We test OIGD against the selected set of baselines for node classification task. We adopt the 5 heterophilic datasets: Cornell, Texas, Wisconsin, Chameleon and Squirrel (Pei et al., 2019). And for homophilic datasets, we adopt

Type	Method	MUTAG	PTC	COX2	PROTEINS	NCI1
Explicit	WL	84.1±1.9	58.0±2.5	83.2±0.2	74.7±0.5	**84.5±0.5**
	DCNN	67.0	56.6	-	61.3	62.6
	DGCNN	85.8	58.6	-	75.5	74.4
	GIN	**89.4±5.6**	64.6±7.0	-	76.2±2.8	82.7±1.7
	FDGNN	88.5±3.8	63.4±5.4	83.3±2.9	76.8±2.9	77.8±1.6
Implicit	IGNN*	76.0±13.4	60.5±6.4	79.7±3.4	76.5±3.4	73.5±1.9
	CGS	**89.4±5.6**	64.7±6.4	-	76.3±4.9	77.6±2.0
	OIGD(ours)	**89.4±6.4**	**66.9±6.6**	**84.8±4.3**	**77.2±2.9**	78.3±1.9

Table 3. Results of graph classification: mean accuracy (%) ± standard deviation over 10 random data splits.

Type	Method	Micro-F1
Explicit	GCN	59.2
	GAT	97.3
	GraphSAGE	78.6
	JKNet	97.6
	GCNII	**99.5**
Implicit	IGNN	97.0
	EIGNN	98.0
	OIGD(ours)	98.4

Table 4. Results of micro-averaged F1 score on PPI dataset.

the 3 citation datasets: Cora, CiteSeer and PubMed. In addition, we also evaluate OIGD on PPI to show that OIGD is applicable to multi-label multi-graph inductive learning. For other datasets except PPI, we adopt the standard data split as Pei et al. (2019) and report the average performance over the 10 random splits. And for PPI, we follow the train/validation/test split used in GraphSAGE (Hamilton et al., 2017).

Baselines. Here we compare our approch against several representative explicit and implicit methods that also adopt the same data split. For explicit models, we select several representative baselines to compare, *i.e.*, GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017), JKNet (Xu et al., 2018), APPNP (Klicpera et al., 2018), Geom-GCN (Pei et al., 2019), GCNII (Chen et al., 2020), and H2GCN (Zhu et al., 2020)). For Geom-GCN, we report the best result among its three model variants. For implicit methods, we present the results of IGNN (Gu et al., 2020) and EIGNN (Liu et al., 2021). We implement IGNN on citation datasets with their 1-layer model used for node classification, and implement EIGNN on citation datasets with their model used for heterophilic datasets. We do not include CGS (Park et al., 2021), as they did not introduce a model for node-level tasks.

Results. As shown in Table 1, our model outperforms the explicit and implicit baselines on all heterophilic datasets by a significant margin, especially on the larger datasets Chameleon and Squirrel. In particular, we improve the current state-of-the-art accuracy of EIGNN from 46.37% to 56.73% on the Squirrel dataset, while the state-of-the-art explicit model H2GCN only reaches 40.75% accuracy. Among explicit models, JKNet, APPNP and GCNII are either designed to consider a larger range of neighbors or to mitigate oversmoothing. Despite that they outperform GCN and GAT, they still perform worse than implicit models with infinitely deep layers (*i.e.*, EIGNN and our OIGD). Since nodes with the same class are far away from each other in heterophilic graphs, the results verify that implicit models are more effective to aggregate information from distant nodes than previous explicit finite aggregation mechanism. Compared to other implicit models (EIGNN and IGNN), OIGD still shows clear advantages. Consequently, we argue that the success of our network stems not only from the implicit setting, but also from our nonlinear diffusion that enhances useful features in aggregation.

Also, our results in Table 2 show that OIGD achieves similar accuracy than the compared methods on 3 citation datasets, although they may not need as many long-range dependencies as the heterophilic datasets. On the PPI dataset, as depicted from Table 4, our OIGD achieves 98.43 micro-averaged F1 score, still superior to other implicit methods and most explicit methods, close to the state-of-the-art models. Moreover, with a small initialization for the parameters, all our training processes are empirically stable.

7.2. Performance on Graph Classification

Datasets. For graph classification, we choose a total of 5 bioinformatics benchmarks: MUTAG, PTC, COX2, NCI1 and PROTEINS (Yanardag & Vishwanathan, 2015). Following identical settings as (Yanardag & Vishwanathan, 2015), we conduct 10-fold cross-validation with LIB-SVM (Chang & Lin, 2011) and report the average prediction accuracy and standard deviations in Table 3.

Baselines. Here, we include the baselines that also have reported results on the chosen datasets. For explicit models, we choose Weisfeiler-Lehman Kernel (WL) (Shervashidze et al., 2011), DCNN (Atwood & Towsley, 2016), DGCNN (Zhang et al., 2018), GIN (Xu et al., 2018) and FDGNN (Gallicchio & Micheli, 2020). For implicit models, we reproduce the result of IGNN (Gu et al., 2020) with their source code for a fair comparison, since it used a different performance metric. We do not include EIGNN (Liu et al., 2021), as they did not introduce a model for graph-level

Reg	Cora	Citeseer	Pubmed
None	88.25±1.25	76.81±1.68	89.22±0.40
$\mathcal{R}_{\text{Lap}}$	**88.39±1.31**	**76.95±1.73**	**89.48±0.34**
$\mathcal{R}_{\text{Dec}}$	88.29±0.92	76.84±1.70	89.28±0.41

Table 5. Comparison of different regularization types.

Method	Preprocessing	Training	Total
IGNN	0	83.67 s	83.67 s
EIGNN	1404.45 s	69.14 s	1473.59 s
CGS	0	118.65 s	118.65 s
OIGD	0	47.31 s	47.31 s

Table 6. Comparison of training time on the Pubmed dataset.

tasks.

Results. In this experiment, OIGD achieves the best performance in 4 out of 5 experiments given the competitive baselines. Such performance further validates OIGD's success that it can still capture long-range dependencies when generalized to unseen testing graphs. Note that OIGD also outperforms both implicit baselines.

7.3. Empirical Understandings of OIGD

Feature regularization. In this experiment, we compare the performance of two kinds of feature regularization: the Laplacian regularization $\mathcal{R}_{\text{Lap}}$ and the feature decorrelation $\mathcal{R}_{\text{Dec}}$. We use the 3 citation dataset for comparison. We choose an appropriate regularization coefficient γ for each dataset. As reported in Table 5, Laplacian regularization improves the performance for the homophilic datasets. We attribute this to the fact that similarity between nodes is an appropriate assumption for these datasets. The feature decorrelation also improves the performance slightly.

Long-range dependency. We follow Gu et al. (2020) and use the synthetic dataset Chains to evaluate models' abilities for capturing long-range dependencies. The goal is to classify nodes in a chain of length l, whose label information is only encoded in the starting end node. We use the identical experimental settings as IGNN and EIGNN. We show in Figure 2 the experimental results with chains of different lengths. In general, the implicit models all outperform the explicit models for longer chains, verifying that they have advantages for capturing long-range dependencies. OIGD and EIGNN both repetitively maintain 100% test accuracy with the length of 200. While EIGNN only applies to linear cases, OIGD can apply to more general nonlinear cases.

Training time. To investigate the training time, we train the 1-layer models on Pubmed dataset for 500 epochs to compare their efficiency. Since we use gradient estimate for training, our model can be faster than those implicit models that compute exact gradients. As shown in Table 6, our model costs much fewer time (47.31 s) than the three models that calculate exact gradients. Among the three

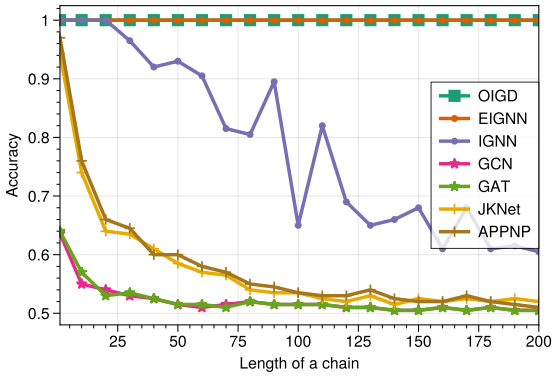

Figure 2. Average accuracy with respect to the length of chains.

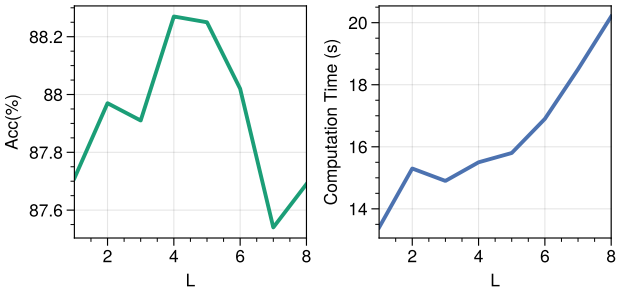

Figure 3. Left: test accuracy (%) with increasing backpropagation steps L on Cora. Right: the corresponding training time (s).

models, EIGNN is the most efficient with 69.14 s training cost, but its pre-processing costs much more than training.

Gradient Estimate. In the left plot of Figure 3, we compare results with different unrolling steps L of Phantom Gradient (Geng et al., 2021) along with their corresponding training time on the Cora dataset. We can see that there is a tradeoff between different L, that either too large or too small L leads to degraded performance, and the sweet spot lies in $L = 4$. The corresponding training time is 15.52 s, which is still preferable to other implicit GNNs (Table 6).

8. Conclusion

In this paper, we develop OIGD, an optimization-induced implicit graph diffusion model, which has access to infinite hops of neighbors while adaptively aggregating features with nonlinear diffusion. We characterize the equilibrium of our implicit layer from an optimization perspective, and show that the learned representation can be formalized as the minimizer of an explicit convex optimization objective. Benefiting from this, we could embed prior properties to the equilibrium and introduce skip connections to promote training stability. Extensive experiments have shown that compared with previous implicit GNNs, our OIGD obtains state-of-the-art performance on various benchmark datasets.

References

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

Attouch, H. and Aze, D. Approximation and regularization of arbitrary functions in hilbert spaces by the lasry-lions method. *Annales de l'Institut Henri Poincaré C, Analyse non linéaire*, 10(3):289–312, 1993.

Atwood, J. and Towsley, D. Diffusion-convolutional neural networks. In *NeurIPS*, 2016.

Ayinde, B. O., Inanc, T., and Zurada, J. M. Regularizing deep neural networks by enhancing diversity in feature extraction. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2650–2661, 2019.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016a.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016b.

Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. In *NeurIPS*, 2019.

Bai, S., Koltun, V., and Kolter, J. Z. Multiscale deep equilibrium models. In *NeurIPS*, 2020.

Broyden, C. G. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19(92):577–593, 1965.

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

Chang, C.-C. and Lin, C.-J. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, 2011.

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

Chung, F. R. and Graham, F. C. *Spectral graph theory*. American Mathematical Soc., 1997.

Eliasof, M., Haber, E., and Treister, E. PDE-GCN: Novel architectures for graph neural networks motivated by partial differential equations. In *NeurIPS*, 2021.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. Graph neural networks for social recommendation. In *WWW*, 2019.

Gallicchio, C. and Micheli, A. Fast and deep graph neural networks. In *AAAI*, 2020.

Gao, H., Chen, Y., and Ji, S. Learning graph pooling and hybrid convolutional operations for text representations. In *WWW*, 2019.

Geng, Z., Zhang, X.-Y., Bai, S., Wang, Y., and Lin, Z. On training implicit models. In *NeurIPS*, 2021.

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

Gribonval, R. and Nikolova, M. A characterization of proximity operators. *Journal of Mathematical Imaging and Vision*, 62(6):773–789, 2020.

Gu, F., Chang, H., Zhu, W., Sojoudi, S., and El Ghaoui, L. Implicit graph neural networks. In *NeurIPS*, 2020.

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

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2016.

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Combining neural networks with personalized pagerank for classification on graphs. In *ICLR*, 2018.

Krantz, S. G. and Parks, H. R. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2012.

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

Liu, J., Kawaguchi, K., Hooi, B., Wang, Y., and Xiao, X. Eignn: Efficient infinite-depth graph neural networks. In *NeurIPS*, 2021.

Nitzberg, M. and Shiota, T. Nonlinear image filtering with edge and corner enhancement. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 14(08):826–833, 1992.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2019.

Pang, J. and Cheung, G. Graph laplacian regularization for image denoising: Analysis in the continuous domain. *IEEE Transactions on Image Processing*, 26(4):1770–1785, 2017.

Park, J., Choo, J., and Park, J. Convergent graph solvers. *arXiv preprint arXiv:2106.01680*, 2021.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. In *ICLR*, 2019.

Perona, P. and Malik, J. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Aanalysis and Machine Intelligence*, 12(7):629–639, 1990.

Rodríguez, P., Gonzàlez, J., Cucurull, G., Gonfaus, J. M., and Roca, X. Regularizing cnns with locally constrained decorrelations. In *ICLR*, 2017.

Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Valsesia, D., Fracastoro, G., and Magli, E. Deep graph-convolutional image denoising. *IEEE Transactions on Image Processing*, 29:8226–8237, 2020.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2017.

Wan, F., Hong, L., Xiao, A., Jiang, T., and Zeng, J. Neodti: neural integration of neighbor information from a heterogeneous network for discovering new drug–target interactions. *Bioinformatics*, 35(1):104–111, 2019.

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

Weickert, J. *Anisotropic diffusion in image processing*. Teubner, 1998.

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

Xie, X., Wang, Q., Ling, Z., Li, X., Wang, Y., Liu, G., and Lin, Z. Optimization induced equilibrium networks. *arXiv preprint arXiv:2105.13228*, 2021.

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

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *SIGKDD*, 2015.

Yang, Y., Liu, T., Wang, Y., Zhou, J., Gan, Q., Wei, Z., Zhang, Z., Huang, Z., and Wipf, D. Graph neural networks inspired by classical iterative algorithms. In *ICML*, 2021.

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

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018.

Zhou, K., Dong, Y., Wang, K., Lee, W. S., Hooi, B., Xu, H., and Feng, J. Understanding and resolving performance degradation in deep graph convolutional networks. In *CIKM*, 2021.

Zhu, H., Lin, Y., Liu, Z., Fu, J., Chua, T.-S., and Sun, M. Graph neural networks with generated parameters for relation extraction. In *ACL*, 2019.

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

Zhu, M., Wang, X., Shi, C., Ji, H., and Cui, P. Interpreting and unifying graph neural networks with an optimization framework. In *WWW*, 2021.

A. Kronecker Product

Given 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 follows:

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} \mathbf{A}_{11}\mathbf{B} & \cdots & \mathbf{A}_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{m1}\mathbf{B} & \cdots & \mathbf{A}_{mn}\mathbf{B} \end{pmatrix}. \tag{14}$$

By definition of the Kronecker product, we have the following important properties of the vectorization with the Kronecker product:

- $\|\mathbf{A} \otimes \mathbf{B}\| = \|\mathbf{A}\| \, \|\mathbf{B}\|$,
- $(\mathbf{A} \otimes \mathbf{B})^\top = \mathbf{A}^\top \otimes \mathbf{B}^\top$,
- $\mathrm{vec}(\mathbf{ABC}) = (\mathbf{C}^\top \otimes \mathbf{A})\mathrm{vec}(\mathbf{B})$.

B. Choice of the Nonlinear Transformation $\sigma(\cdot)$

Oriented Incidence Matrix on Undirected Graphs. If $\mathcal{G}$ is undirected, we randomly assign an orientation for each edge to construct its oriented incidence matrix $\mathbf{G}$, then it is unique to the obtained directed edge set $\tilde{\mathcal{E}}$. Given $\mathbf{U} \in \mathbb{R}^{n\times p}$ as the discrete version of $\mathbf{u}$, the gradient operator $(\mathbf{GU})_{k,\cdot} = \mathbf{u}_j - \mathbf{u}_i$ assigns the edge $k = (i, j) \in \tilde{\mathcal{E}}$ the difference of its endpoint features. Similarly, the divergence operator assigns each node the sum of the features of all edges it shares. Let $F(i, j) = -F(j, i)$, we have

$$(\mathbf{G}^\top \mathbf{F})_{i,\cdot} = \sum_{j:(i,j)\in\mathcal{E}} \mathbf{F}_{(i,j)},$$

where $\mathbf{F} \in \mathbb{R}^{m\times p}$ denotes the feature matrix defined on the edges and $\mathbf{F}_{(i,j)}$ denotes the k-th row of $\mathbf{F}$, where k is the edge that leaves the node i and enters the node j. The two operators are adjoint in the sense that

$$\langle \mathbf{F}, \mathbf{GU} \rangle = \langle \mathbf{G}^\top \mathbf{F}, \mathbf{U} \rangle,$$

where $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{tr}\left(\mathbf{A}^\top \mathbf{B}\right)$ for any equal-size matrices $\mathbf{A}$ and $\mathbf{B}$.

To ensure the discretization of the nonlinear diffusion is well-defined for undirected graphs, we need to ensure that randomly switching the direction of the edge would not influence the output. As a result, we make the following assumption.

Assumption B.1. The nonlinear function $\sigma(\cdot)$ is odd, *i.e.*, $\sigma(-x) = -\sigma(x)$.

Switching the direction of the edge (i, j) to (j, i) is equivalent to multiplying a matrix $\mathbf{E}_{i,j}$ to the left of $\mathbf{G}$, where $\mathbf{E}_{i,j}$ is a diagonal matrix that switches the (i, i)-th element of an identity matrix to -1. If Assumption B.1 holds, we have

$$\begin{aligned} \partial_t \mathbf{U} &= (\mathbf{E}_{i,j}\mathbf{G})^\top \sigma(\mathbf{E}_{i,j}\mathbf{GUK}^\top)\mathbf{K} \\ &= \mathbf{G}^\top \mathbf{E}_{i,j}\mathbf{E}_{i,j}\sigma(\mathbf{GUK}^\top)\mathbf{K} \\ &= \mathbf{G}^\top \sigma(\mathbf{GUK}^\top)\mathbf{K}. \end{aligned}$$

Consequently, the discretization is well-defined for undirected graphs.

Besides Assumption B.1, as discussed in Theorem 4.1, we need Assumption B.2 holds. Overall, $\tanh$ satisfies both assumptions, and has the effect to keep more small-value gradients while shrinking large-value gradients.

Assumption B.2. The nonlinear function $\sigma(\cdot)$ is monotone and L_σ-Lipschitz, *i.e.*,

$$0 \le \frac{\sigma(a) - \sigma(b)}{a - b} \le L_\sigma, \forall a, b \in \mathbb{R}, a \neq b.$$

C. Proofs

C.1. Conditions to be a Proximal Operator

Lemma C.1. *(modified version of Prop. 2 in Gribonval & Nikolova (2020)). Consider $f : \mathcal{H} \to \mathcal{H}$ defined everywhere. The following properties are equivalent:*

(i) there is a proper convex l.s.c *function* $\varphi : \mathcal{H} \to \mathbb{R} \cup \{+\infty\}$ s.t. $f(\mathbf{z}) \in \mathrm{Prox}_{\varphi}(\mathbf{z})$ *for each* $\mathbf{z} \in \mathcal{H}$*;*

(ii) the following conditions hold jointly:

(a) there exists a convex l.s.c *function* $\psi : \mathcal{H} \to \mathbb{R}$ s.t. $\forall \mathbf{y} \in \mathcal{H}, f(\mathbf{y}) = \nabla\psi(\mathbf{y})$*;*
(b) $\|f(\mathbf{y}) - f(\mathbf{y}')\| \leq \|\mathbf{y} - \mathbf{y}'\|, \forall \mathbf{y}, \mathbf{y}' \in \mathcal{H}$.

There exists a choice of $\varphi(\cdot)$ *and* $\psi(\cdot)$*, satisfying (i) and (ii), such that* $\varphi(\mathbf{z}) = \psi^*(\mathbf{z}) - \frac{1}{2}\|\mathbf{z}\|^2$.

Proof. (i) $\Rightarrow$ (ii): Since $\varphi(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}\|^2$ is a proper *l.s.c* 1-strongly convex function, then by Thm. 5.26 in Gribonval & Nikolova (2020), the conjugate function f^* is $\frac{1}{\sigma}$-smooth when f is proper, closed and σ strongly convex and vice versa. Thus, we have:

$$\psi(\mathbf{x}) \coloneqq [\varphi(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}\|^2]^*, \tag{15}$$

is 1-smooth with $\mathrm{dom}(\psi) = \mathcal{H}$. Then we get:

$$f(\mathbf{x}) \in \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u} - \mathbf{x}\|^2 + \varphi(\mathbf{u}) = \{\mathbf{u} : \mathbf{x} \in \partial\varphi(\mathbf{u}) + \mathbf{u}\}, \tag{16}$$

$$= \{\mathbf{u} | \mathbf{x} \in \partial(\varphi(\mathbf{u}) + \frac{1}{2}\|\mathbf{u}\|^2)\}, \tag{17}$$

$$= \{\mathbf{u} | \mathbf{u} = \nabla\psi(\mathbf{x})\} = \{\nabla\psi(\mathbf{x})\}. \tag{18}$$

Hence $f(\mathbf{x}) = \nabla\psi(\mathbf{x})$, and 1-smoothness of ψ implies f is nonexpansive.

(ii) $\Rightarrow$ (i): Let $\varphi(\mathbf{x}) = \psi^*(\mathbf{x}) - \frac{1}{2}\|\mathbf{x}\|^2$. Since $\psi(\mathbf{x})$ is 1-smooth, similarly we can conclude: ψ^* is 1-strongly convex. Hence, φ is convex, and:

$$\mathrm{Prox}_{\varphi}(\mathbf{x}) = \arg\min_{\mathbf{u}}\{\frac{1}{2}\|\mathbf{u} - \mathbf{x}\|^2 + \varphi(\mathbf{u})\}, \tag{19}$$

$$= \{\mathbf{u} | \mathbf{x} \in \partial\varphi(\mathbf{u}) + \mathbf{u}\}, \tag{20}$$

$$= \{\nabla\psi(\mathbf{x})\} = \{f(\mathbf{x})\}, \tag{21}$$

which means $f(\mathbf{x}) = \mathrm{Prox}_{\varphi}(\mathbf{x})$. □

C.2. Proof of Theorem 4.1

The proof follows the one presented in Xie et al. (2021).

Proof. Let $L_\sigma = 1$ and $\left\|\mathbf{K} \otimes \hat{\mathbf{G}}\right\| \leq 1$. Since $\mathbf{1}^\top\tilde{\sigma}(\mathbf{y}) = \sum_{i=1}^n \tilde{\sigma}(y_i)$, we have $\nabla\tilde{\sigma}(\mathbf{y}) = [\sigma(y_1), \cdots, \sigma(y_n)]^\top = \sigma(\mathbf{y})$. By the chain rule, $\nabla\psi(\mathbf{z}) = (\mathbf{K} \otimes \hat{\mathbf{G}})^\top\sigma((\mathbf{K} \otimes \hat{\mathbf{G}})(\mathbf{z} + \mathrm{vec}(b_{\mathbf{\Omega}}(\mathbf{X})))) = f(\mathbf{z})$. Since $\sigma(a)$ is a single-valued function with slope in $[0, 1]$, the element-wise defined operator $\sigma(a)$ is nonexpansive. Combining with $\left\|\mathbf{K} \otimes \hat{\mathbf{G}}\right\| \leq 1$, operator $f(\mathbf{z})$ is also nonexpansive. Due to Lemma C.1, we have $f(\mathbf{z}) = \mathrm{Prox}_{\varphi}(\mathbf{z})$, and $\varphi(\mathbf{z})$ can be chosen as $\psi^*(\mathbf{z}) - \frac{1}{2}\|\mathbf{z}\|^2$. □

D. Row-normalization variant of OIGD

Considering numerical stability, we impose the symmetric normalization to the incidence matrix in our implicit layer (Equation (9a)), which is widely used in GNN models. Alternatively, we provide a row-normalization variant as follows.

$$\mathbf{Z} = \mathbf{D}^{-1}\mathbf{G}^\top\sigma(\mathbf{G}(\mathbf{Z} + b_{\mathbf{\Omega}}(\mathbf{X}))\mathbf{K}^\top)\mathbf{K}. \tag{22}$$

Note that the two formulations are equivalent in the sense that Equation (22) can be rewritten as the following formulation:

$$\bar{\mathbf{Z}} = \hat{\mathbf{G}}^\top\sigma(\hat{\mathbf{G}}(\bar{\mathbf{Z}} + \bar{b}_{\mathbf{\Omega}}(\mathbf{X}))\mathbf{K}^\top)\mathbf{K}, \tag{23}$$

Dataset	Cornell	Texas	Wisconsin	Chameleon	Squirrel	Cora	Citeseer	Pubmed
Nodes	1,283	183	251	2,277	5,201	2,708	3,327	19,717
Edges	280	295	466	31,421	198,493	5429	4732	44338
Features	1,703	1,703	1,703	2,325	2,089	1433	3703	500
Classes	5	5	5	5	5	7	6	3

Table 7. Dataset statistics for node classification task.

Dataset	MUTAG	PTC	COX2	PROTEINS	NCI1
# graphs	188	344	467	1,113	4,110
Avg # nodes	17.9	25.5	41.2	39.1	29.8
Classes	2	2	2	2	2

Table 8. Dataset statistics for graph classification task.

where $\bar{\mathbf{Z}} = \mathbf{D}^{\frac{1}{2}}\mathbf{Z}$ and $\bar{b}_{\mathbf{\Omega}}(\mathbf{X}) = \mathbf{D}^{\frac{1}{2}} b_{\mathbf{\Omega}}(\mathbf{X})$. Moreover, without changing the output $\mathbf{Y}$, the row-normalized version of OIGD has the following formulation:

$$\bar{\mathbf{Z}} = \hat{\mathbf{G}}^{\top}\sigma(\hat{\mathbf{G}}(\bar{\mathbf{Z}} + \bar{b}_{\mathbf{\Omega}}(\mathbf{X}))\mathbf{K}^{\top})\mathbf{K} \tag{24a}$$

$$\mathbf{Y} = g_{\mathbf{\Theta}}(\mathbf{X} + \mathbf{D}^{\frac{1}{2}}\bar{\mathbf{Z}}). \tag{24b}$$

Without loss of generality, all the results can be easily adapted to the row-normalization case.

E. More on Experiments

E.1. Datasets

The statistics for the datasets used in node-level tasks is listed in Table 7. Among heterophilic datasets, Cornell, Texas and Wisconsin are web-page graphs of the corresponding universities, while Chameleon and Squirrel are web-page graphs of Wikipedia of the corresponding topic. The node features are bag-of-word representations of informative nouns in the web-pages. Among homophilic datasets, PPI contains multiple graphs where nodes are proteins and edges are interactions between proteins. The statistics for the datasets used in graph-level tasks is listed in Table 8. All these datasets consist of chemical molecules where nodes refer to atoms while edges refer to atomic bonds.

E.2. Model Architechtures

In terms of model variants, we use symmetric normalized OIGD for the PPI dataset, and row-normalized OIGD for the other datasets. We use a 4-layer model for PPI and a 3-layer model for the two large datasets Chameleon and Squirrel. For the rest datasets, we adopt the model with only one layer. We use linear output function for all the node-level tasks, and MLP for all the graph-level tasks. We adopt the layer normalization (LN) (Ba et al., 2016a) for all the node-level tasks and instance normalization (IN) (Ulyanov et al., 2016) for all the graph-level tasks. They compute the mean and variance used for normalization on a single training case, such that they are independent on the mini-batch size.

E.3. Hyperparamters

In terms of hyperparameters, we tune learning rate, weight decay, α and iteration steps through the Tree-structured Parzen Estimator approach. The hyperparameters for other baselines are consistent with that reported in their papers. Results with identical experimental settings are reused from previous works.