On the Robustness of Graph Neural Diffusion

Anonymous Author(s)
Affiliation
Address
email

Abstract

Neural diffusion on graphs is a novel class of graph neural networks that has attracted increasing attention recently. The capability of graph neural partial differential equations (PDEs) in addressing common hurdles of graph neural networks (GNNs), such as the problems of over-smoothing and bottlenecks, has been investigated but not their robustness to adversarial attacks. In this work, we explore the robustness properties of graph neural PDEs. We empirically demonstrate that graph neural PDEs are intrinsically more robust against adversarial topological perturbations compared to other GNNs. We provide insights into this phenomenon by exploiting the stability of the heat semigroup under graph topology perturbations. We discuss various graph diffusion operators and relate them to existing graph neural PDEs. Furthermore, we propose a general graph neural PDE framework based on which a new class of robust GNNs can be defined. We verify that the new model achieves comparable state-of-the-art performance on several benchmark datasets.

1 Introduction

Deep learning on graphs and Graph Neural Networks (GNNs), in particular, have achieved remarkable success in a variety of application areas such as those related to chemistry (molecules), finance (trading networks) and the social media (the Facebook friend network) [1–5]. GNNs have been applied to various tasks including node classification [3], link prediction [6], and recommender systems [7]. The key to the success of GNNs is the neural message passing scheme [8] where messages are propagated along edges and optimized towards a downstream task.

While the aggregation of neighboring nodes' information is a powerful principle of representation learning, the way that GNNs exchange the information between nodes makes them vulnerable to adversarial attacks [9]. Adversaries can modify the original graph by adding or removing edges [10–12], perturb node attributes [9, 13–15], or inject malicious nodes [16–18]. Small graph perturbations may result in dramatic performance degradation. To defend against adversarial attacks, several robust GNN models have been proposed. Examples include RobustGCN [19], GRAND [20], and ProGNN [21]. In addition, pre-processing based defenders include GNN-SVD [22] and GNNGuard [23].

Recent studies [24–27] have applied neural Ordinary Differential Equations (ODEs) [28] to defend against adversarial attacks. Some works like [24, 27] have revealed interesting intrinsic properties of ODEs that make them more stable than conventional convolutional neural networks (CNNs). Neural Partial Differential Equations (PDEs) have also been applied to graph-structured data [29, 30]. Some papers like [29] (GRAND) and [30] (BLEND) approach deep learning on graphs as a continuous diffusion process and treat GNNs as spatial discretizations of an underlying PDE. However, robustness to adversarial attack has not been studied on such graph neural PDEs. In this work, we demonstrate that graph neural PDEs are intrinsically more robust against adversarial topological perturbations compared to other GNNs. We investigate the heat diffusion on a general Riemannian manifold and show that the diffusion process is essentially stable under small perturbations of the manifold metric.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In doing so, we provide insights into the underlining reasons why graph neural PDEs are stable under graph topological perturbations. Such insights indicate that further improvements to the design of graph neural PDEs are possible. We develop several such improved models under a general graph neural PDE framework and show that these models are also robust to node attribute perturbations.

Main contributions. In this paper, our objective is to develop a general diffusion framework on graphs and study the robustness properties of the induced graph neural PDEs. Our main contributions are summarized as follows:

- We review the notion of heat diffusion on Riemannian manifolds and the stability of its semigroup. We present analogous concepts of gradient, divergence and Laplacian operators for heat diffusion on a graph.
- We generalize heat flow to more general flow schemes, including mean curvature flow and Beltrami flow, which are able to preserve inter-class edges in diffusion. A novel class of graph neural PDEs is thereby induced. We empirically demonstrate that the ability to preserve inter-class edges can boost the robustness of the GNN.
- We show that the proposed graph neural PDEs are stable to graph topology perturbations and node attribute perturbations. Empirical experiments verify that our model achieves comparable state-of-the-art performance on several benchmark datasets.

The rest of this paper is organized as follows. In Section 2, we start with preliminaries on continuous diffusion over a Riemannian manifold and the discrete graph. In Section 3, we present our main results on the stability properties of heat flow on graphs and generalize heat flow to edge-preserving flows from which a new class of graph neural PDEs is proposed. Our proposed model architecture is detailed in Section 4. We summarize experimental results in Section 5 and conclude the paper in Section 6. The proofs for all theoretical results in this paper are given in the supplementary material, where more experiments are also presented.

2 Preliminaries

Similar to [29, 30], we consider a graph as a discretization of a Riemannian manifold. We introduce concepts and notations for flows diffused over a general manifold and a discrete graph. Readers are referred to [31] for more details. The stability of the heat kernel and heat semigroup of the heat diffusion equation under perturbations of the manifold metric is considered. This discussion sheds light on the diffusion stability on graphs under perturbation of the graph topology in the next section.

2.1 Heat Equation and Solution Stability on Manifold

A Riemannian manifold (M, g) is a smooth manifold M endowed with a Riemannian metric g, where the norms of tangent vectors and the angles between them are defined by an inner product. It is the natural generalization of the Euclidean space and correspondingly the divergence, gradient and Laplace operators are well-defined, analogous to their corresponding concepts in Euclidean space. Let $C_0^\infty(M)$ denote the space of smooth functions on M with compact support and $\circ$ the composition operation.

Definition 1. *The* Laplace operator $\Delta : C_0^\infty(M) \mapsto C_0^\infty(M)$ *on a d-dimensional Riemannian manifold (M, g) is defined as*

$$\Delta = \operatorname{div} \circ \nabla,$$

where div *is the* divergence *defined for the C_0^∞ vector fields on M and ∇ is the* gradient *operator. In a local chart U with coordinates $(x^1, x^2, \ldots, x^n)$, for function f and vector field ψ, we have*

$$(\nabla f)^i = \sum_{j=1}^{d} g^{ij} \frac{\partial f}{\partial x^j}, \quad \operatorname{div} \psi = \sum_{j=1}^{d} \frac{1}{\sqrt{\det g}} \frac{\partial}{\partial x^j} \left(\sqrt{\det g} \psi^j \right), \tag{1}$$

where ψ^j is the j-th component of ψ, g^{ij} the components of the inverse metric of g, and det *is the determinant operator of a matrix. We also have*

$$\Delta = \sum_{i,j}^{d} \frac{1}{\sqrt{\det g}} \frac{\partial}{\partial x^i} \left(\sqrt{\det g} g^{ij} \frac{\partial}{\partial x^j} \right). \tag{2}$$

The classical Cauchy problem associated with the heat diffusion equation is to find a function $\varphi(t,x) \in C_0^\infty(\mathbb{R}^+ \times M)$ such that

$$\begin{cases} \frac{\partial \varphi}{\partial t} = \Delta\varphi, \ t > 0, \\ \varphi|_{t=0} = f, \end{cases} \tag{3}$$

where we only consider $f \in C_0^\infty$ and $\mathbb{R}^+$ is the space of positive real numbers.

We can extend the Laplace operator to the generalized *Dirichlet Laplace operator*, denoted as $\mathcal{L} = -\Delta$ and defined on a larger domain[1] [31, section 4.2]. This operator is self-adjoint and non-negative definite on $L^2(M)$, and it can be shown that the above problem (3) is solved by means of the following family $\{P_t\}_{t\geq 0}$ of operators:

$$P_t := e^{-t\mathcal{L}} = \int_{\mathrm{spec}\mathcal{L}} e^{-t\lambda}\,\mathrm{d}E_\lambda = \int_0^\infty e^{-t\lambda}\,\mathrm{d}E_\lambda, \tag{4}$$

where $\{E_\lambda\}_{\lambda\in\mathrm{spec}\mathcal{L}}$ in $L^2(M)$ is the unique *spectral resolution* of the Dirichlet Laplace operator $\mathcal{L}$ and $\mathrm{spec}\mathcal{L}$ is the *spectrum set* of $\mathcal{L}$. The family $\{P_t\}_{t\geq 0}$ is called the *heat semigroup* associated with Δ, and the solution is given by $P_t f$. It is also well-known that the solution of the above problem has an integration form via the (minimal) heat kernel function $k_t(x,y) : \mathbb{R}^+ \times M \times M \mapsto \mathbb{R}$ [31, theorem 7.1]:

$$P_t f(x) = \int_M k_t(x,y) f(y)\,\mathrm{d}\mu(y),$$

where $\mathrm{d}\mu$ is the Riemannian measure [31, section 3.4] on M.

For a Riemannian manifold M, different metrics g give rise to different structures. Let $\widetilde{\Delta}$ be the Laplacian associated with another metric $\tilde{g}$ on M such that, for some $\alpha \geq 1$,

$$\alpha^{-1}\tilde{g} \leq g \leq \alpha\tilde{g}.$$

The new $\widetilde{\Delta}$ is said to be *quasi-isometric* [32] to g, and can be viewed as a *perturbation* of the metric g. $\widetilde{\Delta}$ can be also viewed as a uniformly elliptic operator [32] with respect to (w.r.t.) to g. The stability of the heat semigroup and the heat kernel under perturbations of the Laplace operator (i.e., changes of g) is well studied in [33]. In the special case of uniformly elliptic operators on Euclidean manifold $\mathbb{R}^n$, [33] shows the following theorem.

Theorem 1. *On the Euclidean manifold $\mathbb{R}^n$, let $\{P_t\}_{t\geq 0}$ and $\{\widetilde{P}_t\}_{t\geq 0}$ be two diffusion semigroups on $\mathbb{R}^d$ $(d \geq 2)$ associated with uniformly elliptic operators $\mathrm{div}(A\nabla)$ and $\mathrm{div}(\tilde{A}\nabla)$ with symmetric measurable coefficients $A(x) = (a_{ij}(x))$ and $\tilde{A}(x) = (\tilde{a}_{ij}(x))$, respectively. The corresponding heat kernels are denoted by $p_t(x,y)$ and $\tilde{p}_t(x,y)$. We then have that*

i. There is a bounded, piecewise continuous function $F_1(t,z)$ on $\mathbb{R}^+ \times \mathbb{R}^+$ with $\lim_{z\to 0} F_1(t,z) = 0$ for each $t > 0$ and a constant $c > 0$, both of which depend only on d and α, such that

$$|p_t(x,y) - \tilde{p}_t(x,y)| \leq t^{-d/2} \exp\left(-\frac{\|x-y\|_2^2}{ct}\right) F_1\left(t, \|A - \tilde{A}\|_{L^2_{loc}}\right),$$

for any $(t,x,y) \in \mathbb{R}^+ \times \mathbb{R}^d \times \mathbb{R}^d$, $\|\cdot\|_2$ is the Euclidean norm, and $\|\cdot\|_{L^2_{loc}}$ is the local L^2-norm distance defined in [33].

ii. Furthermore, we also have an L^p-operator norm bound for $P_t - \widetilde{P}_t$ in terms of the local L^2-norm distance between a_{ij} and $\tilde{a}_{ij}$: There is a bounded, piecewise continuous function $F_2(t,z)$ on $\mathbb{R}^+ \times \mathbb{R}^+$ with $\lim_{z\to 0} F_2(t,z) = 0$ for each $t > 0$ that depends only on d and α, such that for any $p \in [1,\infty]$, we have

$$\left\|P_t - \tilde{P}_t\right\|_p \leq F_2\left(t, \|A - \tilde{A}\|_{L^2_{loc}}\right).$$

[1] $\mathcal{L}$ is defined on the larger Sobolev space $W_0^2(M)$ [31], and is the (negative) extension of Δ. We have $\mathcal{L} = -\Delta$ on C_0^∞.

The above theorem shows the pointwise convergence of the heat kernel under perturbations of the matrix A. It proves the stability of the solution under small perturbations of the Laplace operator. These results can also be extended to general Riemannian manifolds [34] since every Riemannian manifold can be isometrically embedded into some Euclidean space [35] and the kernel is isometrically invariant [31, theorem 9.12]. We therefore know that if the difference between $\tilde{A}$ and A (or more generally $\tilde{g}$ and g) are small, the final solutions to (3) have small difference. An intuitive explanation of the stability of the solution comes from the fact that the heat kernel $k_t(x, y)$ is the transition density function of a Brownian motion on the manifold [36] since Brownian motion can be viewed as a diffusion generated by half of the Laplace operator. This means that $k_t(x, y)$ is a weighted average over all possible paths between x and y at time t, which does not change dramatically under small perturbations of g. More specifically, for a Brownian motion on a domain M, if we perturb the g over a subset $P \subset M$ then only the paths passing through P will be affected.

If we go one step further from the uniformly elliptic operators $\operatorname{div}(A\nabla)$ defined in Theorem 1 by extending $A(x)\varphi$ to $A(x, t, \varphi, \nabla\varphi)$, we get the following general *quasilinear parabolic equation*:

$$\begin{cases} \frac{\partial \varphi}{\partial t} = \operatorname{div}(A(x, t, \varphi, \nabla\varphi)), \ t > 0 \\ \varphi|_{t=0} = f, \end{cases} \tag{5}$$

where the principle part $A(x, t, \varphi, \nabla\varphi)$ is equipped with additional structure conditions as shown in [37]. When $A(x, t, \varphi, \nabla\varphi) = A(x, t)$ is a time-dependent diffusion process, analogous bounds to Theorem 1 are shown in [38]. We leave the analysis of general $A(x, t, \varphi, \nabla\varphi)$ for future work.

2.2 Gradient and Divergence Operators on Graphs

We consider a graph as a discretization of a Riemannian manifold. More specifically, consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of a finite set $\mathcal{V}$ of vertices, together with a subset $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ of edges. A graph is weighted when it is associated with a function $w : \mathcal{E} \mapsto \mathbb{R}^+$ which is symmetric, i.e., $w([u, v]) = w([v, u])$, for all $[u, v] \in \mathcal{E}$.

Let $\mathcal{H}(\mathcal{V})$ denote the Hilbert space on $\mathcal{V}$ of real-valued functions with the inner product defined as $\langle a, b\rangle_{\mathcal{H}(\mathcal{V})} = \sum_{v\in\mathcal{V}} a(v)b(v)$, for all $a, b \in \mathcal{H}(\mathcal{V})$. Similarly, we define a Hilbert space $\mathcal{H}(\mathcal{E})$ with inner product $\langle c, d\rangle_{\mathcal{H}(\mathcal{E})} = \sum_{[u,v]\in\mathcal{E}} c([u, v])d([u, v])$, for all $c, d \in \mathcal{H}(\mathcal{E})$. We next define the gradient and divergence operators on graphs analogue to the general continuous manifold defined in Definition 1 as follows [39].

Definition 2. *Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define the following:*

i. The graph gradient is an operator $\nabla : \mathcal{H}(\mathcal{V}) \mapsto \mathcal{H}(\mathcal{E})$ defined by

$$\nabla\varphi([u, v]) = \sqrt{\frac{w([u, v])}{h(v)}}\varphi(v) - \sqrt{\frac{w([u, v])}{h(u)}}\varphi(u), \forall\, [u, v] \in \mathcal{E}, \tag{6}$$

where $h(v) = \sum_{[u,v]\in\mathcal{E}} w([u, v])$ is the degree of node v.

ii. The graph divergence is an operator $\operatorname{div} : \mathcal{H}(\mathcal{E}) \mapsto \mathcal{H}(\mathcal{V})$ defined by

$$(\operatorname{div}\, \psi)(v) = \sum_{[u,v]\in\mathcal{E}} \sqrt{\frac{w([u, v])}{h(v)}}\left(\psi([v, u]) - \psi([u, v])\right). \tag{7}$$

iii. The graph Laplacian is an operator $\Delta : \mathcal{H}(\mathcal{V}) \mapsto \mathcal{H}(\mathcal{V})$ defined by

$$\Delta\varphi = -\frac{1}{2}\operatorname{div}(\nabla\varphi). \tag{8}$$

The graph Laplacian defined above is identical to the normalized Laplacian matrix, i.e.,

$$\Delta = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-1/2}, \tag{9}$$

where $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}(v, v) = h(v)$, and $\mathbf{W}$ is the adjacency matrix satisfying $\mathbf{W}(u, v) = w([u, v])$ if $[u, v] \in \mathcal{E}$ and $\mathbf{W}(u, v) = 0$ otherwise. Note that in (8) we have included the negative sign. The analogue of the graph Laplacian Δ in Section 2.1 is the Dirichlet Laplace operator $\mathcal{L}$. Note the use of the same notation Δ for graphs. The manifold Laplace operator Δ and graph Laplacian Δ will be apparent from the context.

3 Neural Diffusion and Stability on Graphs

We now consider neural diffusion on graphs by making use of concepts from Section 2. Various parabolic-type equations on graphs are studied in this section. The solution stability against adversarial attacks is linked to the stability of the solution for the heat diffusion equation under perturbation of the manifold metric, introduced in Section 2.1. The general framework we consider is based on parabolic-type equations on graphs:

$$\frac{\partial \varphi(u,t)}{\partial t} = \mathrm{div}(A(u,t,\varphi,\nabla\varphi)),\ t>0 \tag{10}$$

with $\varphi(u,0)$ being the initial node attribute at node u and where $A(u,t,\varphi,\nabla\varphi)$ can take different forms. We next provide some examples.

3.1 Continuous Diffusion on Graphs

Definition 3 (Heat diffusion)**.** *The heat diffusion on graphs is defined by*

$$\frac{\partial \varphi(u,t)}{\partial t} = \frac{1}{2}\mathrm{div}(\nabla\varphi)(u,t). \tag{11}$$

Definition 4 (GRAND/BLEND [29, 30])**.** *According to [29, eq (1)] and [30, eq (7) and (9)], the GRAND/BLEND flow is defined by*

$$\frac{\partial \varphi(u,t)}{\partial t} = \frac{1}{2}\mathrm{div}(\nabla_t\varphi)(u,t), \tag{12}$$

where ∇_t is an adaptive Laplace operator (with possible graph rewiring) depending on the evolved node feature $\varphi(\cdot,t)$.

The gradient operator defined in GRAND/BLEND assumes constant edge weight. In Definition 4, the weight function $w([u,v])$ defined in (6) is incorporated into the gradient definition, so it can be absorbed in the time-dependent term $\nabla_t\varphi$. Note that BLEND degenerates to GRAND [29] when there is no positional encoding. In this paper, for fair comparison, we do not use positional encoding for all GNNs.

Analogous to [40], we can define mean curvature flow and Beltrami flow as follows.

Definition 5. *The mean curvature diffusion on graphs is defined by*

$$\frac{\partial \varphi(u,t)}{\partial t} = -\frac{1}{2}\mathrm{div}\left(\frac{\nabla\varphi}{\|\nabla\varphi\|}\right)(u,t), \tag{13}$$

where $-\frac{1}{2}\mathrm{div}\left(\frac{\nabla\varphi}{\|\nabla\varphi\|}\right)$ is a discrete analogue of the mean curvature operator, $\|\nabla\varphi\| = \langle\nabla\varphi,\nabla\varphi\rangle_{\mathcal{H}(E)}^{1/2}$ and $\|\nabla\varphi(u,t)\| = \left(\sum_{[v,u]\in\mathcal{E}}(\nabla\varphi([u,v],t))^2\right)^{1/2}$.

Definition 6. *The Beltrami diffusion on graphs is defined by*

$$\frac{\partial \varphi(u,t)}{\partial t} = -\frac{1}{2}\frac{1}{\|\nabla\varphi\|}\mathrm{div}\left(\frac{\nabla\varphi}{\|\nabla\varphi\|}\right)(u,t). \tag{14}$$

Intuitively, the term $\|\nabla\varphi(u,t)\|$, which appears in (13) and (14) but not in (11), measures the smoothness of the signals in the neighborhood around vertex u at time t. The diffusion using (13) or (14) at vertex u is small when $\|\nabla\varphi(u,t)\|$ is large, i.e., signals are less smooth around vertex u. Hence, mean curvature flow and Beltrami flow are able to preserve the non-smooth graph signals. This phenomenon is visualized in Fig. 1, where mean curvature flow and Beltrami flow are capable of preserving inter-class edges whose weights are much larger than those in heat flow.

3.2 Stability against Graph Topology Perturbation

Attackers can modify the original graph by adding or removing edges [10–12] and perturbing node attributes. Adding or removing edges leads to a different graph Laplacian Δ defined in (8).

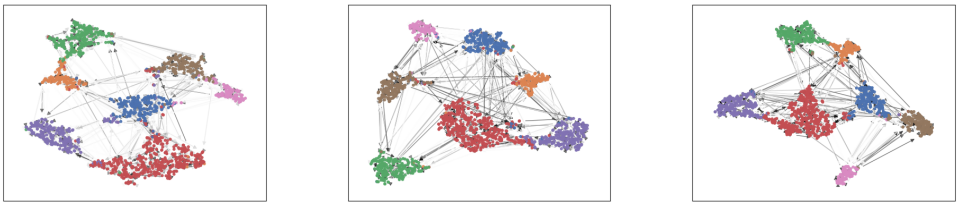

Figure 1: Effect of using different diffusion equations visualized using t-SNE with attention weights. Left: heat flow, middle: mean curvature flow, right: Beltrami flow. The darker the edge, the larger the attention weight.

Consequently, the solution to (11) is altered. Using the language of Section 2.1, the edge perturbations correspond to the perturbation of the metric g (or equivalently A in Theorem 1). According to Theorem 1, the stability of the heat semigroup and the solutions under small perturbations of the Laplace operator is guaranteed. Consequently, the solution is not affected significantly by the edge perturbations. This is different from the notion of stability in [20], where the output $\varphi(u,t)$ is shown to be stable after perturbation of node attributes $\varphi(u,0)$ with fixed graph edges. The stability studied in [20] is called *Lyapunov stability*, which is addressed in Proposition 2. These two notions of stability combined lead to a graph neural PDE in Section 4 that is robust against both edge and node attacks.

The following result shows the stability of the solution of (11), i.e., $\frac{\partial \varphi(u,t)}{\partial t} = -\Delta\varphi(u,t)$, under graph topology or Laplacian perturbation. If the change in the graph topology with respect to any matrix norm is small, the semigroup perturbation can be bounded similarly as the result in Theorem 1. This result considers only the case where the Laplace operator is time-invariant. The time-variant analogy of [38] as discussed after (5) is provided in the supplementary material.

Proposition 1. *Consider* $\Delta = \mathbf{D}^{-1/2}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-1/2}$ *in* (9) *and suppose* $\tilde{\Delta} = \tilde{\mathbf{D}}^{-1/2}(\tilde{\mathbf{D}} - \tilde{\mathbf{W}})\tilde{\mathbf{D}}^{-1/2}$ *with* $\tilde{\mathbf{D}}$ *being the diagonal matrix of* $\tilde{\mathbf{W}} = \mathbf{W} + \mathbf{E}$. *Suppose* $|||\mathbf{E}||| < \varepsilon = o(|||\mathbf{W}|||)$ *for a matrix norm* $|||\cdot|||$, *and* $\mathbf{D}$ *and* $\tilde{\mathbf{D}}$ *are non-singular. Then,* $\|\varphi(u,t) - \tilde{\varphi}(u,t)\| = O(\varepsilon)$.

3.3 Neural Flows on Graphs

Substituting (6) and (7) into (14), we obtain the neural Beltrami flow as

$$\frac{\partial \varphi(u,t)}{\partial t} = \frac{1}{2}\frac{1}{\|\nabla\varphi(u)\|}\sum_{[v,u]\in\mathcal{E}} w([u,v])\left(\frac{1}{\|\nabla\varphi(u)\|} + \frac{1}{\|\nabla\varphi(v)\|}\right)(\varphi(u)-\varphi(v)), \tag{15}$$

where $\|\nabla\varphi(u)\| = \sqrt{\sum_{[v,u]\in\mathcal{E}}(\nabla\varphi([u,v]))^2}$. Here, we further assume $\varphi(u)$, for all nodes u are time-independent for the sake of simplicity. Suppose $|\mathcal{V}| = n$. The function $\varphi : \mathcal{V} \mapsto \mathbb{R}^d$ maps each node to a feature vector. Stacking all the feature vectors together, we obtain $\mathbf{Z} \in \mathbb{R}^{n\times d}$. Let the weight function $w([u,v])$ be the scaled dot product attention function [41] given by

$$w([u,v]) = \mathrm{softmax}\left(\frac{(\mathbf{W}_K\mathbf{z}_u)^\intercal(\mathbf{W}_Q\mathbf{z}_v)}{\sqrt{d_K}}\right), \tag{16}$$

where $\mathbf{W}_K$ and $\mathbf{W}_Q$ are the key and query learnable matrices, respectively, and d_K denotes the number of rows $\mathbf{W}_K$ has. If multi-head attention is applied and denote $\mathbf{A}_h$ as the attention matrix associated with head h, then $\mathbf{A}(\mathbf{Z}) = \frac{1}{h}\sum_h \mathbf{A}_h(\mathbf{Z})$ with $\mathbf{A}_h(u,v) = w_h([u,v])$ where w_h is the weight function for head h. The diffusion equation (15) can be compactly written in matrix form as

$$\frac{\partial \mathbf{Z}(t)}{\partial t} = (\mathbf{A}(\mathbf{Z}(t)) \odot \mathbf{B}(\mathbf{Z}(t)) - \mathbf{\Psi}(\mathbf{Z}(t)))\,\mathbf{Z}(t), \tag{17}$$

where $\odot$ denotes element-wise multiplication, $\mathbf{B}(u,v) = \mathrm{softmax}\left(\frac{1}{\|\nabla\varphi(u)\|^2} + \frac{1}{\|\nabla\varphi(u)\|\|\nabla\varphi(v)\|}\right)$ and $\mathbf{\Psi}(\mathbf{Z}(t))$ is a diagonal matrix with $\Psi(u,u) = \sum_v (\mathbf{A}\odot\mathbf{B})(u,v)$.

Similarly, the diffusion equations using mean curvature flow (13) and heat flow (11) can be written as:

$$\frac{\partial \mathbf{Z}(t)}{\partial t} = (\mathbf{A}(\mathbf{Z}(t)) \odot \mathbf{B}(\mathbf{Z}(t)) - \mathbf{\Psi}(\mathbf{Z}(t)))\,\mathbf{Z}(t), \tag{18}$$

where $\mathbf{B}(u, v) = \mathrm{softmax}\left(\frac{1}{\|\nabla\varphi(u)\|} + \frac{1}{\|\nabla\varphi(v)\|}\right)$ and

$$\frac{\partial \mathbf{Z}(t)}{\partial t} = (\mathbf{A}(\mathbf{Z}(t)) - \mathbf{I})\mathbf{Z}(t), \tag{19}$$

respectively. Although the BLEND model in [30] is inspired from Beltrami flow, its final formulation (equation (8) in [30] where $\mathbf{Z}(t)$ contains positional and node feature embeddings) is indeed (19), i.e., heat flow using attention weight function.

Proposition 2. *Diffusion equations* (17)*,* (18) *and* (19) *are all Lyapunov stable [42].*

4 Model Architecture with Lipschitz Constraint

In this section, we propose a general graph neural PDE network based on the discussion in Section 3. A layer of graph neural PDE is illustrated in Fig. 2. Since we may stack up multiple such layers, the diffusion is performed in a hierarchical manner, where the high-dimensional features are diffused over the graph at the front layers and the low-dimensional features are diffused at the back layers.

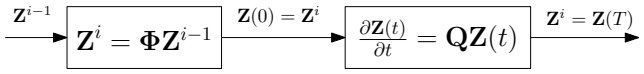

Figure 2: Graph neural PDE at the i-th layer, where each node's feature vector are linearly transformed before being diffused over the graph. We have $\mathbf{Q} := (\mathbf{A}(\mathbf{Z}(t)) \odot \mathbf{B}(\mathbf{Z}(t)) - \mathbf{\Psi}(\mathbf{Z}(t)))$ for mean curvature flow and Beltrami flow, while $\mathbf{Q} := (\mathbf{A}(\mathbf{Z}(t)) - \mathbf{I})$ for heat flow.

One important ingredient in our model is that we perform spectral normalization on $\mathbf{W}_K$ and $\mathbf{W}_Q$ in (16). This is motivated by the fact that attention models have poor performance when the depth increases. In other words, attention weights tend to be uniformly distributed for excessive message exchanges. This phenomenon becomes even more obvious when the nodes' features are diffused according to (17), (18) or (19) as solving such a PDE normally requires many discrete steps. To overcome this over-smoothing problem, we utilize the strategy proposed in [43] to enforce Lipschitz continuity by normalizing the attention scores. Besides, enforcing Lipschitz continuity can also help improve the robustness of the model because the Lipschitz constant controls the perturbation of the output given a bounded input perturbation. Fig. 3 shows that attention weights become overly smooth if no spectral normalization is applied.

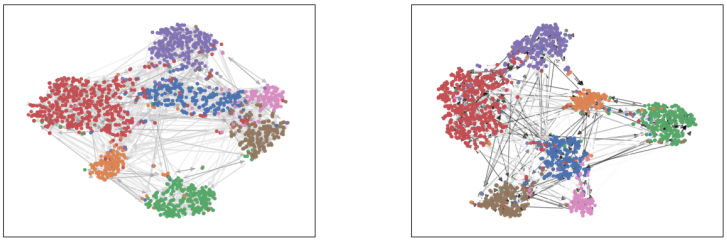

Figure 3: Impact of applying spectral normalization visualized using t-SNE with attention weights using Cora dataset [44]. Left: graph neural PDE with no spectral normalization, right: graph neural PDE with spectral normalization. The darker the edge, the larger the attention weight.

5 Experiments

In this section, we compare graph neural PDEs under different flows to popular GNN architectures: GAT [41], GraphSAGE [45], GIN [46], APPNP [47], and the state-of-the-art GNN defenders:

Table 1: Node classification accuracy (%) on adversarial examples using different GNNs. We denote those experiments that are computationally too heavy to run by "-". The best and the second-best result for each criterion are highlighted in red and blue respectively.

Dataset	Attack	Beltrami in (17)	RobustGCN	GNNGuard	GCNSVD	GAT	GraphSAGE	GIN	APPNP
Cora	*clean*	75.93 ± 1.46	81.34 ± 0.66	79.44 ± 1.18	69.28±1.37	79.74 ± 1.59	76.75 ± 1.52	76.79 ± 1.35	83.06 ± 1.06
	SPEIT	61.87 ± 0.49	36.16 ± 0.41	78.50 ± 2.27	37.50 ± 0.74	38.10 ± 2.48	35.82 ± 0.01	35.82 ± 0.01	36.79 ± 0.61
	TDGIA	62.84 ± 1.17	53.28 ± 8.61	78.92 ± 1.80	40.77 ± 3.34	35.64 ± 12.91	39.78 ± 6.46	39.63 ± 2.38	60.52 ± 4.43
Citeseer	*clean*	70.41 ± 1.38	70.72 ± 1.15	69.69 ± 1.83	66.93 ± 1.07	69.81 ± 1.43	69.78 ± 1.31	68.81 ± 1.58	70.75 ± 0.86
	SPEIT	65.52 ± 2.26	28.56 ± 7.87	69.72 ± 1.84	21.16 ± 1.32	26.00 ± 11.14	19.75 ± 1.82	23.54 ± 5.30	22.19 ± 0.86
	TDGIA	65.77 ± 1.28	38.81 ± 10.84	69.50 ± 1.86	20.77 ± 2.52	19.63 ± 6.53	28.77 ± 7.73	28.65 ± 5.08	54.48 ± 8.56
PubMed	*clean*	86.94 ± 0.25	75.55 ± 0.32	84.80 ± 0.51	-	84.91 ± 0.76	89.22 ± 0.25	76.71 ± 0.14	77.50 ± 0.54
	SPEIT	86.66 ± 0.68	75.54 ± 0.54	84.36 ± 0.58	-	40.94 ± 2.47	39.22 ± 0.00	76.71 ± 0.14	77.55 ± 0.54
	TDGIA	85.56 ± 0.91	75.53 ± 0.36	84.00 ± 1.12	-	39.78 ± 0.29	60.40 ± 11.23	77.58 ± 0.71	77.45 ± 0.68

RobustGCN [19], GNNGuard [23], GCNSVD [22], on standard node classification benchmarks. In our experiments[2], we use the following datasets: Cora (citation networks) [44], Citeseer (citation networks) [48] and PubMed (biomedical literature) [49]. We use a refined version of these datasets provided by [50]. We refer the readers to the supplementary material for more details.

5.1 Attack Setup

We apply the setup introduced in the graph robustness benchmark (GRB) [50]. Based on the assumption that nodes with lower degrees are easier to attack, GRB constructs three test subsets of nodes with different degree distributions. According to the average degrees, GRB defines these subsets of nodes as Easy, Medium, Hard, or Full. In the Easy subset, attacks are easy to succeed and hence the worst performance is expected. For the rest of the nodes, they are divided into a train set (60%) and val set (10%), for training and validation respectively. In our experiments, we choose the Easy subset, i.e., the most challenging mode.

Following GRB, we mainly consider the following real-world adversarial attack settings: *Black-box*: The attacker does not know the defender's method and vice versa. Attackers firstly attack a pre-trained GCN [3] and then transfer the perturbed graphs to the target model. *Evasion*: Attacks will only happen during the inference phase. *Inductive*: GNNs are used to classify unseen data (e.g., new users), i.e., validation or test data are unseen during training. *Injection*: Attackers can only inject new nodes but not modify the target nodes directly. This reflects application scenarios like in online social networks where it is usually hard to hack into users' accounts and modify their profiles. However, it is easier to create fake accounts and connect them to existing users.

Table 2: Node classification accuracy (%) on adversarial examples using BeltramiGuard.

Model	Attack	Cora	Citeseer	PubMed
BeltramiGuard	*clean*	73.01 ± 2.01	69.90 ± 0.44	87.77 ± 0.14
	SPEIT	73.01 ± 2.62	69.90 ± 0.44	87.68 ± 0.03
	TDGIA	72.14 ± 1.20	69.90 ± 0.44	88.16 ± 0.61

We apply two state-of-the-art attack methods: SPEIT [17] and Topological Defective Graph Injection Attack (TDGIA) [18]. These two attacks are the two strongest attacks reported in the GRB. For both attacks, both the maximum allowable injected nodes and the maximum allowable injected edges are set to 50 for Cora and Citeseer, and 300 for PubMed.

Table 1 shows that graph neural PDE induced from Beltrami flow is more robust than all other GNNs except for GNNGuard, which is specifically designed to remove malicious edges and is thus robust against topology perturbation. So, our graph neural PDE can be simply combined with GNNGuard which we denote as BeltramiGuard. Table 2 shows that BeltramiGuard makes attacks' efforts totally in vain and exceeds GNNGuard in Citeseer and PubMed datasets. Note, our vanilla Beltrami itself can surpass GNNGuard on the large PubMed. More details are shown in supplementary material.

[2]Our experiments are run on a GeForce RTX 3090 GPU.

5.2 Ablation Studies

Diffusion Schemes

We compare different diffusion equations: heat flow (19) where $w([u, v]) = 1$ for all $[u, v] \in \mathcal{E}$, GRAND/BLEND [29, 30] which is another heat flow where $w([u, v])$ in (19) is the attention function defined in (16), mean curvature flow (18), and Beltrami flow (17). For GRAND/BLEND, we stack three neural PDE layers using the architecture proposed in Fig. 2 since the original GRAND/BLEND in [29, 30], which has only one PDE layer, does not perform well.

Table 3: Node classification accuracy (%) on adversarial examples using graph neural PDEs induced from different flows, where implicit Adam PDE solver with step size 2 is used.

Dataset	Attack	Heat	GRAND/BLEND	Mean Curvature	Beltrami
Cora	*clean*	78.86 ± 1.78	74.89 ± 1.29	76.01 ± 2.20	75.93 ± 1.46
	SPEIT	38.66 ± 2.40	55.67 ± 3.60	60.67 ± 1.31	61.87 ± 0.49
	TDGIA	60.26 ± 5.19	59.55 ± 6.41	62.01 ± 2.37	62.84 ± 1.17
Citeseer	*clean*	69.47 ± 1.22	69.45 ± 0.91	70.50 ± 1.63	70.41 ± 1.38
	SPEIT	22.95 ± 5.07	39.94 ± 6.94	65.39 ± 1.62	65.52 ± 2.26
	TDGIA	52.42 ± 11.2	54.92 ± 7.64	66.83 ± 1.59	65.77 ± 1.28
PubMed	*clean*	86.31 ± 0.46	88.89 ± 0.38	88.45 ± 0.32	86.94 ± 0.25
	SPEIT	40.77 ± 1.78	39.39 ± 0.26	87.13 ± 0.33	86.66 ± 0.68
	TDGIA	42.63 ± 5.28	63.91 ± 11.61	85.79 ± 0.82	85.56 ± 0.91

From Table 3, we can observe that 1) the flows which are capable of preserving non-smooth features are generally more robust than heat flow, 2) the proposed mean curvature flow and Beltrami flow are more robust than GRAND/BLEND, and 3) the two flows we proposed generally suffer less performance variance than the other methods.

PDE Solvers

The impact of PDE solvers on the performance can be observed in Table 4. The adaptive step-size solver Dopri5 performs better than the fixed-step solvers (Implicit/Explicit Adam) at the cost of higher computational complexity. For fixed-step solvers, increasing the step size τ reduces the variance. For sufficiently large step sizes, the implicit method converges faster than the explicit method.

Table 4: Node classification accuracy (%) using graph neural PDEs induced from Beltrami flow, when different PDE solvers are applied. Experiments are conducted on Citeseer dataset.

PDE solvers	Param.	Clean	SPEIT
Implicit Adam	$\tau = 1$	70.59 ± 2.26	64.64 ± 2.60
	$\tau = 10$	70.22 ± 0.70	64.14 ± 1.00
Explicit Adam	$\tau = 1$	69.72 ± 1.14	62.88 ± 2.70
	$\tau = 10$	69.91 ± 0.96	64.76 ± 1.34
Dopri5	-	70.91 ± 0.98	66.96 ± 1.46

6 Conclusion

In this paper, we have introduced a general graph neural PDE framework from which several graph neural PDEs are proposed. We analyzed the robustness of the graph neural PDEs and showed that graph neural PDEs are inherently robust against topology perturbations and are also Lyapunov stable. We provided theoretical evidence showing that the robustness of graph neural PDEs stems from the stability of the heat kernel and semigroup during the diffusion process. Moreover, we conducted extensive experiments that empirically verify the robustness of the proposed graph neural PDEs when compared with an existing graph neural PDE induced from approximated heat flow, popular GNNs and the state-of-the-art GNN defenders.

References

[1] X. Yue, Z. Wang, J. Huang, S. Parthasarathy, S. Moosavinasab, Y. Huang, S. M. Lin, W. Zhang, P. Zhang, and H. Sun, "Graph embedding on biomedical networks: methods, applications and evaluations," *Bioinformatics*, vol. 36, no. 4, pp. 1241–1251, 2019.

[2] H. Ashoor, X. Chen, W. Rosikiewicz, J. Wang, A. Cheng, P. Wang, Y. Ruan, and S. Li, "Graph embedding and unsupervised learning predict genomic sub-compartments from hic chromatin interaction data," *Nat. Commun.*, vol. 11, 2020.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.

[4] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 249–270, Jan 2022.

[5] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, 2021.

[6] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proc. Advances Neural Inf. Process. Syst. Workshop*, 2016.

[7] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. Int. Conf. Knowledge Discovery and Data Mining*, 2018.

[8] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*, 2017.

[9] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2018.

[10] J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan, "Fast gradient attack on network embedding," *ArXiv*, 2018.

[11] M. Waniek, T. P. Michalak, M. J. Wooldridge, and T. Rahwan, "Hiding individuals and communities in a social network," *Nature Human Behaviour*, vol. 2, no. 1, pp. 139–147, 2018.

[12] J. Du, S. Zhang, G. Wu, J. M. F. Moura, and S. Kar, "Topology adaptive graph convolutional networks," *ArXiv*, vol. abs/1710.10370, 2017.

[13] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *Proc. Int. Conf. Learn. Representations*, 2019.

[14] Y. Ma, S. Wang, T. Derr, L. Wu, and J. Tang, "Graph adversarial attack via rewiring," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2021, p. 1161–1169.

[15] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar, "Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach," in *Proc. Web Conf.*, 2020, p. 673–683.

[16] J. Wang, M. Luo, F. Suya, J. Li, Z. Yang, and Q. Zheng, "Scalable attack on graph data by injecting vicious nodes," *Data Mining Knowl. Discovery*, pp. 1 – 27, 2020.

[17] Q. Zheng, Y. Fei, Y. Li, Q. Liu, M. Hu, and Q. Sun. Kdd cup 2020 ml track 2 adversarial attacks and defense on academic graph 1st place solution. Accessed: May 1, 2022. [Online]. Available: https://github.com/Stanislas0/KDD_CUP_2020_MLTrack2_SPEIT

[18] X. Zou, Q. Zheng, Y. Dong, X. Guan, E. Kharlamov, J. Lu, and J. Tang, "Tdgia: Effective injection attacks on graph neural networks," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2021, p. 2461–2471.

[19] D. Zhu, Z. Zhang, P. Cui, and W. Zhu, "Robust graph convolutional networks against adversarial attacks," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2019, p. 1399–1407.

[20] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang, "Graph random neural networks for semi-supervised learning on graphs," in *Proc. Advances Neural Inf. Process. Syst.*, 2020.

[21] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2020, p. 66–74.

[22] N. Entezari, S. A. Al-Sayouri, A. Darvishzadeh, and E. E. Papalexakis, "All you need is low (rank): Defending against adversarial attacks on graphs," in *Proc. Int. Conf. Web Search Data Mining*, 2020, p. 169–177.

[23] X. Zhang and M. Zitnik, "Gnnguard: Defending graph neural networks against adversarial attacks," in *Proc. Advances Neural Inf. Process. Syst.*, 2020.

[24] H. Yan, J. Du, V. Y. Tan, and J. Feng, "On robustness of neural ordinary differential equations," in *Proc. Advances Neural Inf. Process. Syst.*, 2018, pp. 1–13.

[25] E. Haber and L. Ruthotto, "Stable architectures for deep neural networks," *Inverse Problems*, vol. 34, no. 1, pp. 1–23, Dec. 2017.

[26] X. Liu, S. Si, Q. Cao, S. Kumar, and C.-J. Hsieh, "How does noise help robustness? Explanation and exploration under the neural sde framework," in *Proc. Conf. Comput. Vision Pattern Recognition*, 2020, pp. 282–290.

[27] Q. Kang, Y. Song, Q. Ding, and W. P. Tay, "Stable neural ODE with Lyapunov-stable equilibrium points for defending against adversarial attacks," in *Proc. Advances Neural Inf. Process. Syst.*, 2021.

[28] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *arXiv preprint arXiv:1806.07366*, 2018.

[29] B. P. Chamberlain, J. Rowbottom, M. Goronova, S. Webb, E. Rossi, and M. M. Bronstein, "Grand: Graph neural diffusion," in *Proc. Int. Conf. Mach. Learn.*, 2021.

[30] B. P. Chamberlain, J. Rowbottom, D. Eynard, F. Di Giovanni, D. Xiaowen, and M. M. Bronstein, "Beltrami flow and neural diffusion on graphs," in *Proc. Advances Neural Inf. Process. Syst.*, 2021.

[31] A. Grigoryan, *Heat kernel and analysis on manifolds*. Providence: American Mathematical Soc., 2009.

[32] L. Saloff-Coste, "Uniformly elliptic operators on riemannian manifolds," *J. Differ. Geometry*, vol. 36, no. 2, pp. 417–450, 1992.

[33] Z.-Q. Chen, Z. Qian, Y. Hu, and W. Zheng, "Stability and approximations of symmetric diffusion semigroups and kernels," *J. functional anal.*, vol. 152, no. 1, pp. 255–280, 1998.

[34] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," *Computer graphics forum*, vol. 28, no. 5, pp. 1383–1392, 2009.

[35] J. Nash, "The imbedding problem for riemannian manifolds," *Ann. math.*, vol. 63, no. 1, pp. 20–63, 1956.

[36] E. P. Hsu, *Stochastic analysis on manifolds*. Providence: American Mathematical Soc., 2002.

[37] E. DiBenedetto, U. P. Gianazza, and V. Vespri, *Harnack's inequality for degenerate and singular parabolic equations*. London: Springer Science & Business Media, 2011.

[38] W. Zheng, "Stability of time-dependent diffusion semigroups and kernels," *Acta Math. Sinica*, vol. 15, no. 4, pp. 575–586, 1999.

[39] D. Zhou and B. Schölkopf, "Regularization on discrete spaces," in *Joint Pattern Recognition Symposium*. Springer, 2005, pp. 361–368.

[40] N. Sochen, R. Kimmel, and R. Malladi, "A general framework for low level vision," *IEEE Trans. Image Process.*, vol. 7, no. 3, pp. 310–318, 1998.

[41] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–12.

[42] C.-T. Chen and B. Shafai, *Linear system theory and design*. New York: Oxford university press New York, 1999.

[43] G. Dasoulas, K. Scaman, and A. Virmaux, "Lipschitz normalization for self-attention layers with application to graph neural networks," in *Proc. Int. Conf. Mach. Learning*, 2021, pp. 2456–2466.

[44] A. McCallum, K. Nigam, J. D. M. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, pp. 127–163, 2004.

[45] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Advances Neural Inf. Process. Syst.*, 2017.

[46] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. Int. Conf. Learn. Representations*, 2019.

[47] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *Proc. Int. Conf. Learning Representations*, 2019.

[48] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, p. 93, Sep. 2008.

[49] G. M. Namata, B. London, L. Getoor, and B. Huang, "Query-driven active surveying for collective classification," in *Workshop Mining and Learn. with Graphs*, 2012.

[50] Q. Zheng, X. Zou, Y. Dong, Y. Cen, D. Yin, J. Xu, Y. Yang, and J. Tang, "Graph robustness benchmark: Benchmarking the adversarial robustness of graph machine learning," *Neural Information Processing Systems Track on Datasets and Benchmarks 2021*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] . We have left more complicated problems to future work.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] . See the Broader Impact section in supplementary material.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] . See Section 2 and Section 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] . See supplementary material.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] . See supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] . See Section 5 and supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] . We run each experiment for 10 times.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] . See Section 5.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5 for the open-source datasets and models we have used from GitHub.
   (b) Did you mention the license of the assets? [No] . Please see the licenses given in the GitHub link or datasets websites.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] . No new assets.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] . The datasets used in the paper are all open-source datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] . There is no identifiable information or offensive content in the datasets.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]