On the Robustness of Graph Neural Diffusion to Topology Perturbations

Yang Song*
Nanyang Technological University
C3.ai
yang.song@c3.ai

Qiyu Kang*
Nanyang Technological University
kang0080@e.ntu.edu.sg

Sijie Wang*
Nanyang Technological University
wang1679@e.ntu.edu.sg

Kai Zhao*
Nanyang Technological University
kai.zhao@ntu.edu.sg

Wee Peng Tay
Nanyang Technological University
wptay@ntu.edu.sg

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 topology perturbation as 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 toward a downstream task.

While aggregating neighboring nodes' information is a powerful principle of representation learning, the way that GNNs exchange information between nodes makes them vulnerable to adversarial attacks [9]. Adversaries can perturb a graph's topology by adding or removing edges [10–12] or by injecting malicious nodes to the original graph [13–15]. Another common attack is to perturb node attributes [9, 16–18]. Our paper will mainly tackle graph topology perturbation. Adversaries who can

*Equal contribution.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

inject nodes to the original graph while not modifying the original graph directly are called injection attacks [13–15]. Adversaries who can directly modify the original graph including edges and node features are called modification attacks [10–12, 9, 16–18]. 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. 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 show that the proposed graph neural PDEs are intrinsically robust to graph topology perturbations. We verify that the new 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 of 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 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[2] [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:

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

where $\{E_\lambda\}_{\lambda \in \operatorname{spec}\mathcal{L}}$ in $L^2(M)$ is the unique *spectral resolution* of the Dirichlet Laplace operator $\mathcal{L}$ and $\operatorname{spec}\mathcal{L}$ 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 7.13]:

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

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 $\mathbb{R}^d$, the uniformly elliptic operator can be written as $\operatorname{div}(A\nabla)$ with symmetric measurable coefficients $A(x) = (a_{ij}(x))$ such that $\sum_{i,j} a_{ij}(x)\xi_i\xi_j \geq \lambda_0 |\xi|^2$ for all $\xi \in \mathbb{R}^d$ and for some constant $\lambda > 0$ independent of x. The reference [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* $\operatorname{div}(A\nabla)$ *and* $\operatorname{div}(\tilde{A}\nabla)$ *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*

[2] $\mathcal{L}$ 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^∞.

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

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

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 $\mathrm{div}(A\nabla)$ defined in Theorem 1 by extending $A(x)\varphi$ to $A(x,t,\varphi,\nabla\varphi)$, we get the following general *quasilinear parabolic equation*:

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

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 $\mathrm{div} : \mathcal{H}(\mathcal{E}) \mapsto \mathcal{H}(\mathcal{V})$ *defined by*

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

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

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

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 a 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,* $\|\nabla\varphi\| = \langle\nabla\varphi,\nabla\varphi\rangle^{1/2}_{\mathcal{H}(E)}$ *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.

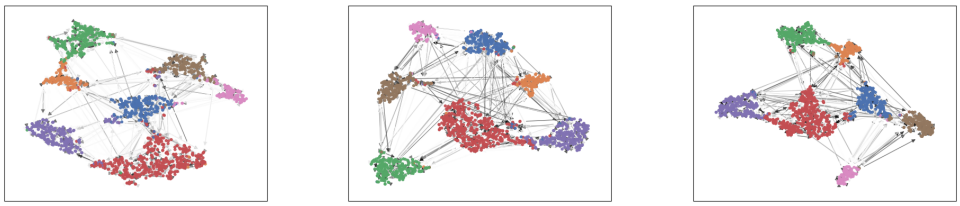

Figure 1: Effect of using different diffusion equations visualized using t-SNE with attention weights. Left: heat flow, middle: mean curvature flow, right: Beltrami flow. The darker the edge, the larger the attention weight.

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). 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 degree matrix defined analogously to* (9) *for* $\tilde{\mathbf{W}} = \mathbf{W} + \mathbf{E}$. *Suppose* $\varepsilon := |||\mathbf{E}||| = o(1)$ *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

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

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

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

where $\odot$ denotes element-wise multiplication, $\mathbf{B}(u, v) = \operatorname{softmax}\left(\frac{1}{\|\nabla\varphi(u)\|^2} + \frac{1}{\|\nabla\varphi(u)\|\|\nabla\varphi(v)\|}\right)$ 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:

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

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

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 illustrates that attention weights become overly smooth if no spectral normalization is applied.

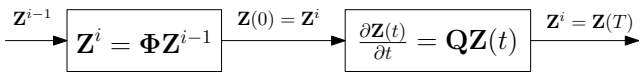

Figure 2: Graph neural PDE at the i-th layer, where each node's feature vector are linearly transformed before being diffused over the graph. We have $\mathbf{Q} = \mathbf{A}(\mathbf{Z}(t)) \odot \mathbf{B}(\mathbf{Z}(t)) - \mathbf{\Psi}(\mathbf{Z}(t))$ for mean curvature flow and Beltrami flow, while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

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: Robust-GCN [19], GNNGuard [23], GCNSVD [22], on standard node classification benchmarks. In our experiments,[3] we use the following datasets: Cora (citation networks) [44], Citeseer (citation networks)

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

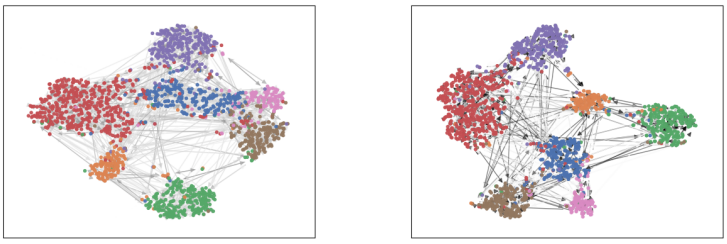

Figure 3: Impact of applying spectral normalization visualized using t-SNE with attention weights using Cora dataset [44]. Left: graph neural PDE with no spectral normalization, right: graph neural PDE with spectral normalization. The darker the edge, the larger the attention weight.

[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. Our experiment codes are provided in `https://github.com/zknus/Robustness-of-Graph-Neural-Diffusion`.

Table 1: Node classification accuracy (%) on adversarial examples using different GNNs. The implicit Adam PDE solver with step size 2 is used for Beltrami. 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*	70.14 ± 1.80	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	66.46 ± 1.33	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

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. The remaining nodes 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, in this paper, 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 applications 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. For other attack settings such as white-box attacks and modification attacks, relevant experiments and discussions are provided in the supplementary material.

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 injection attack methods: SPEIT [14] and Topological Defective Graph Injection Attack (TDGIA) [15]. 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. We provide more details about how these two attacks work under evasion, black-box, and injection setting in the supplemental material.

Table 1 indicates 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 injection attacks. This suggests that the output features from a graph PDE are stable under topology perturbations, as suggested by Proposition 1.

Furthermore, the heat diffusion process on a manifold tends to diffuse differently under different geometries. For example, it diffuses slower at points with positive curvature, and faster at points with negative curvature [34, 51]. In [52] and [53], the authors show different datasets have different geometric properties like hyperbolicity distribution or Balanced Forman curvature. Our theoretical analysis only shows a loose uniform bound in terms of the adjacency matrix, while the performance is very much dataset-dependent. More advanced theoretical analysis for different datasets is highly non-trivial and needs further investigations.

Our graph neural PDE can be combined with GNNGuard. We denote this model as BeltramiGuard. Table 2 demonstrates that BeltramiGuard renders attacks in vain and exceeds GNNGuard on the Citeseer and PubMed datasets. Note from Table 1 that the vanilla Beltrami model already surpasses GNNGuard on the large PubMed dataset. More details are provided in the supplementary material.

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	70.14 ± 1.80
	SPEIT	22.95 ± 5.07	39.94 ± 6.94	65.39 ± 1.62	66.46 ± 1.33
	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 observe that even the vanilla time-invariant heat flow preserves some robustness as compared to non-PDE GNNs in Table 1. This further validates our theoretical analysis in Proposition 1, which suggests that if the topology perturbation is bounded, the learned representations are close to those under the "clean" scenario. 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) these two flows generally suffer less performance variance than the other methods.

More Experiments

Due to space constraint, we refer the reader to the supplementary material for more ablation studies including the impact of the Lipschitz constraint, PDE solvers, the number of layers, and time complexity. The model performance under other attacks is also presented in the supplementary material.

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.

7 Acknowledgement

This research is supported by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE-T2EP20220-0002 and A*STAR under its RIE2020 Advanced Manufacturing and Engineering (AME) Industry Alignment Fund – Pre Positioning (IAF-PP) (Grant No. A19D6a0053).

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 *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] 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.

[14] 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

[15] 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.

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

[17] 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.

[18] 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.

[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 *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 *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 *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 *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 *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 *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.

[51] D. Bhaskar, K. MacDonald, D. Thomas, S. Zhao, K. You, J. Paige, Y. Aizenbud, B. Rieck, I. M. Adelstein, and S. Krishnaswamy, "Diffusion-based methods for estimating curvature in data," in *ICLR Workshop Geometrical Topological Representation Learn.*, 2022.

[52] J. Topping, F. D. Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein, "Understanding over-squashing and bottlenecks on graphs via curvature," in *Proc. Int. Conf. Learn. Representations*, 2022.

[53] S. Zhu, S. Pan, C. Zhou, J. Wu, Y. Cao, and B. Wang, "Graph geometry interaction learning," in *Advances Neural Inf. Process. Syst.*, 2020.

[54] J. McAuley and J. Leskovec, "Image labeling on a network: using social-network metadata for image classification." in *Proc. European Conf. Comput. Vision*, 2012.

[55] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," *Relational Representation Learning Workshop, NeurIPS*, 2018.

[56] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-based recommendations on styles and substitutes," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, 2015, p. 43–52.

[57] C. Van Loan, "The sensitivity of the matrix exponential," *SIAM Journal on Numerical Analysis*, vol. 14, no. 6, pp. 971–981, 1977.

[58] R. A. Horn and C. R. Johnson, *Matrix analysis*. New York: Cambridge university press, 2012.

[59] D. Hinrichsen, A. Ilchmann, and A. J. Pritchard, "Robustness of stability of time-varying linear systems," *J. Differ. Equ.*, vol. 82, no. 2, pp. 219–250, 1989.

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]

Appendix

In Appendix A of this supplementary material, we provide more details about the datasets used in our main paper. We briefly describe the attack settings in Appendix B. More experiments and ablation studies that are not included in the main paper due to space constraints are now presented in Appendix C. We include supplemental experiments with more datasets in Appendix D. Further ablation studies of our model are also presented in Appendix E. In addition, we present an extension of mean curvature and Beltrami flows in Appendix F. Additional implementation details of the models are provided in Appendix G. The proofs for all theoretical results in the main paper are given in Appendix H.

A Datasets

In our main paper, we conduct experiments using the first three datasets in Table 4: Cora (citation networks) [44], Citeseer (citation networks) [48] and PubMed (biomedical literature) [49]. In this supplementary material, we conduct further experiments using the other three datasets in Table 4: Flickr (social networks) [54], Coauthor (academic networks) [55] and Amazon Computer (recommendation networks) [56]. We use a refined version of these datasets provided by [50], where the main statistics are summarized in Table 4. The features in these datasets are normalized by an $\arctan$ (bijective) transformation [50], which permits attackers to restore the original features and thus allows real-world adversarial attacks.

Table 4: Statistic of Datasets

Dataset	# Nodes	# Edges	# Features	# Classed	Feature Range (norm)
Cora	2,680	5,148	302	7	$-0.94 \sim 0.94$
Citeseer	3,191	4,172	768	6	$-0.96 \sim 0.89$
PubMed	19,717	44,325	500	3	$-0.14 \sim 0.99$
Coauthor	18,333	81,894	6,805	15	$-0.04 \sim 1.00$
Flickr	89,250	449,878	500	7	$-0.47 \sim 1.00$
Amazon Computer	13,752	245,861	767	10	$-0.40 \sim 0.60$

B Attack Settings

There are three common categories of adversarial attacks studied in the literature [50]:

- Poison (attack occurs in training) or evasion (attack occurs in testing).
- White-box (attackers knows target model/method) or black-box (attackers do not know target model/method and thus need to attack a surrogate model and then transfer to target model).
- Injection (attackers inject nodes/edges to the original graph and generate attributes for the injected nodes) or modification (attackers modify the original graph including its topology and node features directly).

Our experiments in the main paper mainly focus on the evasion, black-box, and injection attack setting, which we believe are the most realistic attack settings. The two selected attack methods SPEIT and TDGIA are tailored for these attack settings. To be more specific, we carry out the following:

- Evasion: SPEIT and TDGIA are performed on a trained model during testing time.
- Black-box: SPEIT and TDGIA are used to attack a trained GCN, i.e., a surrogate model, to generate graph perturbations, and then the target model is tested on this perturbed graph.
- Injection: when perturbing the graph based on a trained GCN, SPEIT and TDGIA first inject new nodes into the original graph and then generate the injected nodes' features.

In this supplementary material, we include more attack settings such as white-box attacks and modification attacks. Further experiments to test graph PDE robustness against node attribute perturbation are also included.

C More Experiments and Ablation Studies

We include more experiments and ablation studies that are not included in the main paper due to space constraints. More specially, in this section, we conduct experiments to study the effect of the Lipschitz constraint and the number of layers, inference time complexity, more injection attacks including evasion white-box injection attacks and attacks with various attack strengths, and the modification attackers which modify the original graph including its topology and node features directly.

Our codes are developed based on the following two repositories:

- `https://github.com/twitter-research/graph-neural-pde` and
- `https://github.com/THUDM/grb`,

where the new diffusion schemes and their induced neural PDEs are developed based on the first repository and we follow the second repository to set up the robustness evaluation benchmark.

C.1 PDE Solvers

The impact of PDE solvers on the performance can be observed in Table 5. 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 5: 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	TDGIA
Implicit Adam	$\tau = 1$	70.59 ± 2.26	64.64 ± 2.60	65.62 ± 0.96
	$\tau = 2$	70.14 ± 1.80	66.46 ± 1.33	65.77 ± 1.28
	$\tau = 10$	70.22 ± 0.70	64.14 ± 1.00	65.75 ± 1.65
Explicit Adam	$\tau = 1$	69.72 ± 1.14	62.88 ± 2.70	64.81 ± 1.62
	$\tau = 2$	69.59 ± 0.92	65.05 ± 1.39	65.38 ± 1.35
	$\tau = 10$	69.91 ± 0.96	64.76 ± 1.34	65.52 ± 3.17
Dopri5	-	70.91 ± 0.98	66.96 ± 1.46	67.01 ± 1.94

C.2 Lipschitz Constraint and Number of Layers

On one hand, keeping as few layers as possible can mitigate the over-smoothing problem. On the other hand, too few layers result in the underfitting problem, i.e., the test clean accuracy is low. To understand this better, we performed experiments using a different number of layers. The results are summarized in Table 6. The number of layers is a hyperparameter we tune during training.

We have tried using the same number of layers for GRAND/BLEND as in their paper. However, the test clean accuracy is low. This is mainly due to two reasons: 1) we are using the inductive setting whereas the paper of GRAND/BLEND uses transductive training; 2) the datasets we are using have been calibrated for the robustness evaluation, which is different from the original datasets used by the paper of GRAND/BLEND. For example, grb-cora [50] has node feature size of 302 while the original Cora dataset has node feature size of 1433. Table 6 provides empirical evidence for GRAND/BLEND needing more layers in our setting.

C.3 Time Complexity

We have summarized the time complexity in Table 7. The time is computed by averaging 500 diffusion operations. We can see that Beltrami and GRAND/BLEND have similar time complexity to the two

Table 6: Node classification accuracy (%) on adversarial examples generated from SPEIT. We apply graph neural PDEs induced from Beltrami flow with or without the Lipschitz constraint. Experiments are conducted on Cora dataset.

Clean/Robust acc.	# layers used	Beltrami	Beltrami w/o Lips.
Clean	1	63.96 ± 1.95	63.51 ± 3.18
	2	72.46 ± 0.76	69.93 ± 1.48
	3	73.51 ± 1.49	72.69 ± 1.89
	4	75.93 ± 1.46	75.15 ± 1.41
Robust	1	59.10 ± 3.20	58.06 ± 1.78
	2	63.58 ± 1.39	58.96 ± 2.82
	3	60.52 ± 1.56	57.46 ± 1.65
	4	61.87 ± 0.49	56.79 ± 1.52

defenders GNNGuard and GCNSVD small step sizes but incur more computation time than the other GNNs to complete a diffusion process. Heat, unlike Beltrami and GRAND/BLEND, does not use attention and has the lowest time complexity among all the neural PDEs.

Table 7: Top: Average time spent on a Beltrami diffusion process, i.e., time to solve (17) and a counterpart in GRAND/BLEND, when different PDE solvers are applied and multiple step size options are tested for each solver. Bottom: Average time spent on an aggregation step using different GNNs. Experiments are conducted on the Citeseer dataset.

PDE solvers	Param.	Beltrami	GRAND/BLEND	Heat
Implicit Adam	$\tau = 1$	9.8ms	6.6ms	3.0ms
	$\tau = 2$	17.0ms	11.2ms	4.0ms
	$\tau = 10$	48.1ms	46.6ms	9.2ms
Explicit Adam	$\tau = 1$	10.0ms	6.8ms	3.0ms
	$\tau = 2$	16.8ms	11.2ms	3.6ms
	$\tau = 10$	32.6ms	21.0ms	5.8ms
Dopri5	-	66.0ms	20.0ms	13.0ms

RobustGCN	GNNGuard	GCNSVD	GAT	GraphSAGE	GIN	APPNP
0.6ms	13.2ms	9.0ms	1.8ms	0.8ms	1.0ms	1.6ms

C.4 White-Box Attacks

We now include the white-box attacks for our model and the baselines. The results are shown in Table 8. We observe that our model outperforms the baselines under white-box attacks. The results further validate that neural PDEs are intrinsically robust to graph topology perturbations, as indicated by Proposition 1 and Lemma 1.

C.5 Injection Attacks with Various Attack Strengths

Under injection attacks with a different number of injection nodes and edges, we have performed more experiments with various attack strengths. Table 9 shows that the robustness is not significantly affected by the number of nodes and edges injected by attackers as the original is not perturbed in this setting.

C.6 Modification Attacks

In the modification attack, attackers can directly flip the original graph's edges and perturb the features of the nodes. We apply the PGD method to randomly flip edges and then perturb node features. In Table 10, we observe that

Table 8: Node classification accuracy (%) on adversarial examples generated from *white-box* attacks. The best and the second-best result for each criterion are highlighted in red and blue, respectively.

Dataset	Attack	BeltramiGuard	Beltrami	RobustGCN	GNNGuard	GCNSVD	GAT	GraphSAGE	GIN	APPNP
Cora	*clean*	73.01 ± 2.01	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	71.94 ± 0.31	48.95 ± 3.32	36.12 ± 0.31	80.22 ± 0.91	33.06 ± 7.86	19.18 ± 9.47	17.16 ± 11.30	21.49 ± 13.08	19.77 ± 10.39
	TDGIA	67.39 ± 1.97	52.61 ± 4.20	36.27 ± 0.61	78.43 ± 1.10	11.94 ± 0.0	7.99 ± 4.40	21.57 ± 4.82	38.06 ± 4.69	51.94 ± 5.32
Citeseer	*clean*	69.90 ± 0.44	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	68.78 ± 0.82	55.24 ± 6.90	20.19 ± 2.61	69.22 ± 1.90	19.31 ± 3.55	14.67 ± 5.05	19.81 ± 3.06	12.54 ± 6.33	20.75 ± 2.32
	TDGIA	67.96 ± 1.22	53.61 ± 14.01	18.68 ± 4.06	68.40 ± 1.44	16.93 ± 2.41	14.23 ± 5.05	20.69 ± 5.64	18.87 ± 3.61	25.70 ± 6.78
PubMed	*clean*	87.77 ± 0.14	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	85.26 ± 1.42	85.13 ± 0.79	75.07 ± 0.30	84.30 ± 1.34	-	39.22 ± 0.0	40.46 ± 1.69	75.58 ± 1.03	77.62 ± 0.10
	TDGIA	81.36 ± 3.09	84.88 ± 0.46	75.78 ± 0.32	83.33 ± 2.91	-	37.96 ± 1.82	44.85 ± 3.06	75.72 ± 0.70	77.32 ± 0.44

Table 9: Node classification accuracy (%) on adversarial examples generated from SPETI and TDGIA under black-box injection setting where a different number of injected nodes and edges are applied.

Dataset	Attack	# nods/edges injected	Beltrami	RobustGCN	GNNGuard	GCNSVD	GAT	GraphSAGE	GIN	APPNP
Cora	SPEIT	50/50	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
	SPEIT	100/100	57.18 ± 1.68	35.82 ± 0.00	79.39 ± 1.41	35.82 ± 0.00	37.22 ± 1.87	35.82 ± 0.00	35.82 ± 0.00	35.82 ± 0.00
	SPEIT	150/150	55.88 ± 0.56	35.82 ± 0.00	80.78 ± 0.71	35.82 ± 0.00	26.12 ± 16.29	35.82 ± 0.00	35.82 ± 0.00	29.85 ± 11.94
	SPEIT	200/200	58.21 ± 2.81	35.82 ± 0.00	79.38 ± 0.64	29.88 ± 11.96	29.85 ± 11.94	35.82 ± 0.00	35.82 ± 0.00	22.30 ± 15.83
	TDGIA	50/50	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
	TDGIA	100/100	62.44 ± 1.56	54.11 ± 3.18	77.99 ± 2.24	42.29 ± 0.43	36.94 ± 16.96	34.24 ± 9.99	37.03 ± 1.59	62.50 ± 3.47
	TDGIA	150/150	64.18 ± 0.87	52.43 ± 10.84	79.39 ± 1.74	11.94 ± 0.00	39.74 ± 16.09	38.34 ± 1.48	35.82 ± 0.00	61.37 ± 7.07
	TDGIA	200/200	61.48 ± 1.68	48.88 ± 6.60	80.78 ± 1.89	11.94 ± 0.00	46.74 ± 12.70	38.34 ± 2.50	36.19 ± 0.75	61.10 ± 0.98
Citeseer	SPEIT	50/50	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
	SPEIT	100/100	64.74 ± 1.27	17.01 ± 6.05	69.51 ± 1.61	19.75 ± 2.56	19.12 ± 1.47	18.81 ± 2.74	20.61 ± 2.39	20.53 ± 1.57
	SPEIT	150/150	65.13 ± 2.27	20.69 ± 1.49	70.22 ± 1.63	18.18 ± 1.81	18.97 ± 1.57	18.34 ± 1.63	19.75 ± 0.00	19.75 ± 0.00
	SPEIT	200/200	64.11 ± 0.83	15.52 ± 7.19	69.36 ± 1.29	18.96 ± 3.00	19.12 ± 1.26	22.10 ± 5.11	21.16 ± 4.97	17.71 ± 1.37
	TDGIA	50/50	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
	TDGIA	100/100	65.60 ± 1.68	45.61 ± 6.75	66.22 ± 5.44	25.39 ± 6.33	25.08 ± 10.21	35.81 ± 11.99	29.94 ± 7.26	52.74 ± 6.48
	TDGIA	150/150	66.22 ± 2.31	39.66 ± 15.80	69.36 ± 0.90	24.14 ± 3.21	19.36 ± 2.14	26.57 ± 9.28	27.51 ± 6.69	59.01 ± 3.93
	TDGIA	200/200	64.66 ± 2.15	38.48 ± 9.63	69.91 ± 1.54	26.49 ± 4.14	22.33 ± 11.29	26.72 ± 13.41	26.73 ± 7.47	51.72 ± 5.06
PubMed	SPEIT	300/300	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
	SPEIT	600/600	86.80 ± 0.98	75.71 ± 0.60	86.47 ± 0.58	-	34.8 ± 10.99	41.10 ± 1.63	76.36 ± 0.65	77.22 ± 0.43
	SPEIT	900/900	87.10 ± 0.38	76.00 ± 0.35	86.73 ± 0.48	-	39.99 ± 1.55	40.78 ± 1.77	76.73 ± 0.36	77.14 ± 0.70
	SPEIT	1200/1200	87.51 ± 0.58	74.75 ± 0.90	86.19 ± 0.40	-	40.82 ± 1.85	40.08 ± 1.63	76.70 ± 0.72	77.17 ± 0.32
	TDGIA	300/300	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
	TDGIA	600/600	86.33 ± 0.84	75.84 ± 0.23	84.21 ± 0.33	-	56.00 ± 9.13	70.24 ± 20.07	76.38 ± 0.75	77.37 ± 0.31
	TDGIA	900/900	87.00 ± 1.47	75.91 ± 0.55	83.87 ± 1.01	-	41.44 ± 10.61	52.40 ± 6.03	75.56 ± 0.58	76.88 ± 0.68
	TDGIA	1200/1200	86.87 ± 1.41	75.62 ± 0.82	84.35 ± 2.24	-	57.16 ± 10.92	71.72 ± 14.02	76.49 ± 0.80	77.23 ± 0.46

1) the robustness performance starts to break down when the 60% of nodes have their features perturbed by $\epsilon = 0.1$ (the value of features is in [-1,1]) and 60% of edges are flipped, and
2) as long as feature perturbation is small, the robustness can still be retained even if 80% of nodes have their features perturbed and 80% of edges are flipped.

In summary, from those extensive experiment results, we observe that our models are more robust against topology perturbation than feature perturbation.

Table 10: Node classification accuracy (%) on adversarial examples generated from PGD under black-box *modification* setting where a different number of modified nodes and edges are applied. Experiments are conducted on Cora dataset.

Ratio of nodes/edges modified	Feature perturbation	Beltrami
20%/20%	$\epsilon = 0.01$	73.73 ± 0.86
40%/40%	$\epsilon = 0.01$	73.28 ± 1.36
60%/60%	$\epsilon = 0.01$	72.46 ± 1.48
80%/80%	$\epsilon = 0.01$	72.31 ± 2.46
20%/20%	$\epsilon = 0.1$	61.56 ± 1.06
40%/40%	$\epsilon = 0.1$	52.15 ± 2.31
60%/60%	$\epsilon = 0.1$	44.03 ± 0.79
80%/80%	$\epsilon = 0.1$	40.30 ± 0.91
80%/80%	$\epsilon = 1$	39.55 ± 3.46
80%/80%	$\epsilon = 2$	37.09 ± 3.02
80%/80%	$\epsilon = 5$	37.16 ± 1.88
80%/80%	$\epsilon = 10$	37.16 ± 0.33

C.7 More Discussion about Neural Heat Diffusion

"Heat" in Table 3 of the main paper serves as a baseline. Recall that both "Heat" and GRAND/BLEND are derived from heat flow, where GRAND/BLEND uses the attention function to weigh edges while "Heat" uses the constant function, which is not learned from the data and hence is not expected to perform well. In a further experiment, we now treat the constant in "Heat" as a trainable variable that is shared by all edges. We denote this new variant as Heat$^+$ and compare it with GAT and APPNP (which were more robust than "Heat") under SPEIT attack in Table 11. We now see that Heat$^+$ is more robust. This constant variable controls the diffusivity on the graph, analogous to the thermal diffusivity on a manifold. It is interesting to observe such a phenomenon since the experiment indicates that heat diffusivity also affects robustness. We believe further investigations in this direction can be performed in future work.

Table 11: Node classification accuracy (%) on adversarial examples generated from SPEIT.

Dataset	GNN	Clean	10/10	15/15	20/20	25/25	30/30	35/35	40/40	45/45	50/50
Citeseer	Heat$^+$	69.78 $\pm$ 0.95	69.09 $\pm$ 1.61	67.77 $\pm$ 1.61	66.27 $\pm$ 2.41	66.27 $\pm$ 2.81	64.26 $\pm$ 1.27	61.88 $\pm$ 3.01	57.30 $\pm$ 4.07	54.61 $\pm$ 5.25	54.61 $\pm$ 2.92
	GAT	69.81 $\pm$ 1.43	60.56 $\pm$ 4.77	62.01 $\pm$ 4.89	45.52 $\pm$ 10.44	30.34 $\pm$ 5.40	31.16 $\pm$ 6.16	31.03 $\pm$ 12.60	36.11 $\pm$ 13.85	23.13 $\pm$ 3.25	26.00 $\pm$ 11.14
	APPNP	70.75 $\pm$ 0.86	68.65 $\pm$ 0.91	66.52 $\pm$ 2.90	59.87 $\pm$ 3.49	56.11 $\pm$ 9.70	56.05 $\pm$ 2.37	51.22 $\pm$ 8.88	38.50 $\pm$ 8.75	31.60 $\pm$ 9.80	22.19 $\pm$ 0.86

D Further Experiments

We repeat the experiments in Table 1 in the main paper but using the three new datasets. The inherent robustness of the proposed graph neural PDEs is again verified by Table 12, especially on the Flickr and Coauthor datasets.

Table 12: Node classification accuracy (%) on adversarial examples generated by the SPEIT method. 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	RobustGCN	GNNGuard	GCNSVD	GAT	GraphSAGE	GIN	APPNP
Flickr	*clean*	49.40 $\pm$ 0.12	47.66 $\pm$ 0.00	-	-	54.45 $\pm$ 0.57	53.50 $\pm$ 0.02	53.57 $\pm$ 0.29	54.08 $\pm$ 0.14
	SPEIT	49.79 $\pm$ 0.68	6.57 $\pm$ 0.00	-	-	6.57 $\pm$ 0.00	49.71 $\pm$ 0.00	49.71 $\pm$ 0.00	6.57 $\pm$ 0.00
	TDGIA	49.47 $\pm$ 0.50	52.95 $\pm$ 1.58	-	-	50.38 $\pm$ 0.20	50.21 $\pm$ 0.11	50.14 $\pm$ 0.25	54.28 $\pm$ 0.59
Coauthor	*clean*	95.83 $\pm$ 0.30	87.75 $\pm$ 0.23	92.56 $\pm$ 0.16	-	92.75 $\pm$ 0.15	94.53 $\pm$ 0.21	84.91 $\pm$ 0.32	87.67 $\pm$ 0.16
	SPEIT	94.83 $\pm$ 0.12	87.62 $\pm$ 0.29	92.56 $\pm$ 0.16	-	2.59 $\pm$ 1.46	39.44 $\pm$ 13.97	39.44 $\pm$ 13.97	87.66 $\pm$ 0.16
	TDGIA	95.06 $\pm$ 0.21	87.3 $\pm$ 0.29	-	-	65.32 $\pm$ 13.04	87.97 $\pm$ 3.72	85.12 $\pm$ 0.33	87.54 $\pm$ 0.13
Amazon Computer	*clean*	87.86 $\pm$ 0.30	86.22 $\pm$ 0.54	88.77 $\pm$ 0.05	74.79 $\pm$ 0.68	89.21 $\pm$ 0.60	89.92 $\pm$ 0.33	86.44 $\pm$ 0.23	82.66 $\pm$ 1.54
	SPEIT	84.90 $\pm$ 0.61	86.33 $\pm$ 0.62	88.62 $\pm$ 0.05	26.79 $\pm$ 1.25	26.88 $\pm$ 16.68	29.19 $\pm$ 9.35	86.44 $\pm$ 0.23	82.62 $\pm$ 1.55
	TDGIA	85.50 $\pm$ 0.15	86.69 $\pm$ 0.51	-	-	55.45 $\pm$ 23.07	63.14 $\pm$ 10.59	86.65 $\pm$ 0.57	83.44 $\pm$ 1.59

E Further Ablation Studies

We repeat the experiments in Table 3 of the main paper but using three new datasets. Similar to what we have stated in the main paper, we observe the advantage of the proposed mean curvature flow and Beltrami flow in terms of robustness against adversarial attacks.

F Extension

The graph Laplacian and curvature can be generalized to an operator that can be thought of as the discrete analog of the p-Laplacian in the continuous case [39, 37]:

$$\frac{\partial \varphi(u,t)}{\partial t} = \frac{1}{2}\mathrm{div}\left(\|\nabla\varphi\|^{p-2}\nabla\varphi\right)(u,t), \tag{20}$$

where $p = 1, 2$ correspond to the mean curvature and heat (GRAND/BLEND) equations, respectively. Here, we consider the cases where $p > 2$. Note that when $p > 2$, as opposed to the cases where $p \leq 1$, e.g., $p = 1$ for mean curvature and Beltrami flows, the edges that connect nodes with similar features are potentially preserved in the diffusion process. Table 14 shows that the PDE's robustness decreases as p increases.

Table 13: 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	Beltrami	Mean Curvature	GRAND/BLEND	Heat
Coauthor	*clean*	95.81 ± 0.38	95.66 ± 0.18	94.35 ± 0.33	92.87 ± 0.55
	SPEIT	95.10 ± 0.35	95.41 ± 0.28	66.63 ± 9.76	34.18 ± 8.85
	TDGIA	95.06 ± 0.21	95.41 ± 0.17	85.97 ± 4.50	62.39 ± 15.96
Amazon Computer	*clean*	87.86 ± 0.30	87.59 ± 0.44	90.59 ± 0.35	90.59 ± 0.57
	SPEIT	84.90 ± 0.61	84.55 ± 1.03	75.91 ± 12.22	46.82 ± 14.42
	TDGIA	85.50 ± 0.15	85.34 ± 0.78	86.31 ± 2.50	76.60 ± 10.48
Flickr	*clean*	49.42 ± 0.12	47.66 ± 1.51	48.52 ± 0.19	46.65 ± 1.30
	SPEIT	49.76 ± 0.63	49.74 ± 1.18	49.75 ± 0.06	49.70 ± 0.02
	TDGIA	49.47 ± 0.50	47.39 ± 1.43	48.63 ± 0.09	47.05 ± 1.20

Table 14: Node classification accuracy (%) on adversarial examples using graph neural PDEs induced from p-Laplacian flow where $p = 3, 4$, where implicit Adam PDE solver with step size 2 is used. All experiments are done on Citeseer dataset.

Attack	p-Laplacian ($p = 3$)	p-Laplacian ($p = 4$)	Mean Curvature	Beltrami
clean	69.15 ± 1.37	67.27 ± 1.14	70.50 ± 1.63	70.41 ± 1.38
SPEIT	63.76 ± 0.85	62.44 ± 1.14	65.39 ± 1.62	65.52 ± 2.26
TDGIA	64.89 ± 0.77	63.39 ± 2.00	66.83 ± 1.59	65.77 ± 1.28

G Implementation Details

By default, all the models are implemented using three layers, with layer normalization, 50% dropout at the end of each layer, and hidden feature dimensions $64 - 64$. Here are some notes that should be taken:

- When dealing with the Cora dataset, PDEs induced by Beltrami and mean curvature flows tend to underfit. Thus, we implement them using four layers with hidden feature dimensions $128 - 128 - 128$.
- For all neural PDEs, node features are diffused independently over layers, i.e., each layer solves its own PDE. At each layer, once a diffusion process is over, the parameters in the associated PDE are reset. This operation is effective to alleviate the overfitting problem.
- By default, the integral period for all PDEs is set to $[0, 1]$.

All experiments are repeated 5 to 10 times with different random seeds.

For all the attacks, the maximum number of nodes and edges that are allowed to be perturbed are summarized in Table 15.

Table 15: Statistic of attacks' budgets on each dataset

Dataset	max # Nodes	max # Edges
Cora	50	50
Citeseer	50	50
PubMed	300	300
Coauthor	150	300
Amazon Computer	100	200
Flickr	1000	5000

GNNGuard implemented in Table 1 of the main paper is based on GCN, where each layer in GNNGuard contains two operations: 1) adjusting attention coefficients by pruning likely fake edges

and assigning less weight to suspicious edges; and 2) a normal GCN layer. Regarding BeltramiGuard implemented in Table 2 of the main paper, we replace the second component as mentioned above, i.e., a GCN layer, with a PDE layer induced by Beltrami flow.

H Proofs of Results

In this section, we provide detailed proof of the results stated in the main paper. The time-variant analogy of [38] as discussed after (5) in the main paper is also presented.

H.1 Proof of Proposition 1

For the system $\frac{\partial \varphi(u,t)}{\partial t} = -\Delta\varphi(u,t)$, the solution is given by $\varphi(u,t) = e^{-t\Delta}\varphi(u,0)$. We therefore need to derive a bound for $|||e^{-t\Delta} - e^{-t\tilde{\Delta}}|||$. Note for matrix exponent, in general $e^{X+Y} \neq e^X e^Y$ unless X and Y commute (i.e. $XY = YX$). However, from [57, eq (3.5)], we have

$$\left|\left|\left|e^{-t\Delta} - e^{-t\tilde{\Delta}}\right|\right|\right| \leq t\left|\left|\left|\Delta - \tilde{\Delta}\right|\right|\right| \left|\left|\left|e^{-t\Delta}\right|\right|\right| \left|\left|\left|e^{|||t(\Delta-\tilde{\Delta})|||}\right|\right|\right|,$$

where the positive definiteness of Δ is used. We further analyze $\Delta - \tilde{\Delta}$. Since $\tilde{\mathbf{W}} = \mathbf{W} + \mathbf{E}$ with $|||\mathbf{E}||| = \varepsilon$, we have $|||\mathbf{D} - \tilde{\mathbf{D}}||| = O(\varepsilon)$ because norms for a finite dimensional space are equivalent [58, Theorem 5.4.4.]. Let $\mathbf{E}' = \tilde{\mathbf{D}}^{-1/2} - \mathbf{D}^{-1/2}$. We also have $|||\mathbf{E}'||| = O(\varepsilon)$.

It follows that

$$\begin{aligned}
\left|\left|\left|\Delta - \tilde{\Delta}\right|\right|\right| &= \left|\left|\left|\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} - \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{W}}\tilde{\mathbf{D}}^{-1/2}\right|\right|\right| \\
&= \left|\left|\left|\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} - (\mathbf{D}^{-1/2} + \mathbf{E}')(\mathbf{W} + \mathbf{E})(\mathbf{D}^{-1/2} + \mathbf{E}')\right|\right|\right| \\
&= \left|\left|\left|\mathbf{E}'(\mathbf{W} + \mathbf{E}')(\mathbf{D}^{-1/2} + \mathbf{E}') + \mathbf{D}^{-1/2}\mathbf{E}(\mathbf{D}^{-1/2} + \mathbf{E}') + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{E}'\right|\right|\right| \\
&= O(\varepsilon).
\end{aligned}$$

We therefore obtain the conclusion that $|||e^{-t\Delta} - e^{-t\tilde{\Delta}}||| = O(\varepsilon t e^{-\rho t})$ for some constant $\rho > 0$ since $|||e^{-t\Delta}||| = O(e^{-t\rho})$ according to [57] and the fact that Δ is positive definite according to our assumption. The conclusion $\|\varphi(u,t) - \tilde{\varphi}(u,t)\| = O(\varepsilon t e^{-\rho t})$ then follows.

H.2 Proof of Proposition 2

Proof. Since $\mathbf{A}$ is right stochastic, all eigenvalues of $\mathbf{A} - \mathbf{I}$ have non-positive real parts. By the Lyapunov Stability Theorem [42], the stability of (19) is ensured. Since $\mathbf{\Psi}^{-1}\mathbf{A} \odot \mathbf{B}$ is right stochastic, all eigenvalues of $\mathbf{\Psi}^{-1}\mathbf{A} \odot \mathbf{B} - \mathbf{I}$ have non-positive parts and thus (18) and (17) are stable. □

H.3 The Time-Variant Case

In this section, we provide an analysis of the time-variant system as alluded to before Proposition 1. In the time-variant system, we have

$$\frac{\partial \varphi(u,t)}{\partial t} = -\Delta(t)\varphi(u,t), \tag{21}$$

where $\Delta(t)$ is a time-variant Laplacian operator. Let the state transition matrix be $\Phi_\Delta(t;0)$ so that the solution of (21) is given by $\varphi(u,t) = \Phi_\Delta(t;0)\,\varphi(u,0)$ [42].

In [59], the authors discuss the stability radius of time-variant systems. The system still preserves stability if the perturbation of $\Delta(t)$ is smaller than a stability radius. In this discussion, we assume the perturbation is small, i.e., it is inside the stability radius so that both $\Delta(t)$ and its perturbed version $\tilde{\Delta}(t)$ generate exponentially stable solutions. In other words, for a given matrix norm $|||\cdot|||$, there exist positive constants M, $\tilde{M}$, ω and $\tilde{\omega}$ such that

$$|||\Phi_\Delta(t,s)||| \leq Me^{-\omega(t-s)}, \text{ and } |||\Phi_{\tilde{\Delta}}(t,s)||| \leq \tilde{M}e^{-\tilde{\omega}(t-s)}, \quad t \geq s \geq 0. \tag{22}$$

To simplify the analysis, we further assume the perturbation is small enough such that the exponent difference is also small, i.e.,

$$|\omega - \tilde{\omega}| < \omega. \tag{23}$$

We now consider the solution difference after perturbation. Let $\mathbf{D}(t)$ and $\mathbf{W}(t)$ be the time-variant version of $\mathbf{D}$ and $\mathbf{W}$, respectively. They are assumed to satisfy (22) and (23).

Lemma 1. *Let* $\Delta(t) = \mathbf{D}^{-1/2}(t)(\mathbf{D}(t) - \mathbf{W}(t))\mathbf{D}^{-1/2}(t)$ *and* $\tilde{\Delta}(t) = \tilde{\mathbf{D}}^{-1/2}(t)(\tilde{\mathbf{D}}(t) - \tilde{\mathbf{W}}(t))\tilde{\mathbf{D}}^{-1/2}(t)$ *with* $\tilde{\mathbf{D}}(t)$ *being the diagonal degree matrix for* $\tilde{\mathbf{W}}(t) = \mathbf{W}(t) + \mathbf{E}(t)$*, where* $\{\tilde{\mathbf{W}}(t) : t \geq 0\}$ *satisfies* (22) *and* (23)*. Denote* $\varepsilon(t) = |||\mathbf{E}(t)|||$*. We have* $\|\varphi(u,t) - \tilde{\varphi}(u,t)\| = O\Big(e^{-\rho t}\int_0^t \varepsilon(\tau)\,\mathrm{d}\tau\Big)$ *for some constant* $\rho > 0$*.*

Proof. For the perturbed system, we have

$$\begin{aligned}\frac{\partial \varphi(u,t)}{\partial t} &= -\tilde{\Delta}(t)\varphi(u,t) \\ &= -\Delta(t)\varphi(u,t) + \Big(\Delta(t) - \tilde{\Delta}(t)\Big)\,\varphi(u,t).\end{aligned} \tag{24}$$

Formally, (24) can be interpreted as a closed loop system obtained by applying the dynamical feedback $\Big(\Delta(t) - \tilde{\Delta}(t)\Big)\,\varphi(u,t)$. According to [42], the solution of the above system, denoted by $\tilde{\varphi}(u,t)$, satisfies

$$\begin{aligned}\tilde{\varphi}(u,t) &= \Phi_\Delta\,(t,0)\,\varphi(u,0) + \int_0^t \Phi_\Delta(t,\tau)\,\Big(\Delta(\tau) - \tilde{\Delta}(\tau)\Big)\,\tilde{\varphi}(u,\tau)\,\mathrm{d}\tau \\ &= \varphi(u,t) + \int_0^t \Phi_\Delta(t,\tau)\,\Big(\Delta(\tau) - \tilde{\Delta}(\tau)\Big)\,\tilde{\varphi}(u,\tau)\,\mathrm{d}\tau.\end{aligned}$$

Similar to the proof in Appendix H.1 for the time-invariant case, we have $|||\Delta(\tau) - \tilde{\Delta}(\tau)||| = O(\varepsilon(t))$. It follows that

$$\begin{aligned}\|\tilde{\varphi}(u,t) - \varphi(u,t)\| &= \left\|\int_0^t \Phi_\Delta(t,\tau)\,\Big(\Delta(\tau) - \tilde{\Delta}(\tau)\Big)\,\tilde{\varphi}(u,\tau)\,\mathrm{d}\tau\right\| \\ &= \left\|\int_0^t \Phi_\Delta(t,\tau)\,\Big(\Delta(\tau) - \tilde{\Delta}(\tau)\Big)\,\Phi_{\tilde{\Delta}}(\tau,0)\varphi(u,0)\,\mathrm{d}\tau\right\| \\ &\leq \int_0^t |||\Phi_\Delta(t,\tau)|||\left|\left|\left|\Delta(\tau) - \tilde{\Delta}(\tau)\right|\right|\right| |||\Phi_{\tilde{\Delta}}(\tau,0)|||\,\mathrm{d}\tau \cdot \|\varphi(u,0)\| \\ &\leq \int_0^t M e^{-\omega(t-\tau)}\tilde{M} e^{-\tilde{\omega}(\tau-0)}\varepsilon(\tau)\,\mathrm{d}\tau \cdot \|\varphi(u,0)\| \\ &= O\Big(e^{-\rho t}\int_0^t \varepsilon(\tau)\,\mathrm{d}\tau\Big)\end{aligned}$$

for some constant $\rho > 0$. In the second equality, we have used the fact that the initial point for both the unperturbed and perturbed system is $\varphi(u,0)$, and the last equality follows from (22) and (23). □

Remark H.1. *For the sake of simplicity, in this work, we only present theoretical analysis for the time-invariant or time-variant graph Laplacian in Proposition 1 and Lemma 1 for heat diffusion* (11)*. The more complicated case where the graph Laplacian* Δ *depends on node features* u *as implemented in* (16) *is left for future work. From experiment results, for example, the results shown in Table 3 in the main paper and Table 13 in this supplementary material, the time-invariant case already preserve some robustness as compared to non-PDE GNNs in Table 1 and Table 12, which validates our theoretical analysis. The more robust mean curvature flow and Beltrami flow proposed in the paper are highly non-linear, making a theoretical analysis difficult. However, our analysis for the time-variant case provides some insights as to why robustness is present in these cases, as demonstrated in our experiments.*

Broader Impact

Our work develops robust GNNs to mitigate the threat of adversarial attacks, which can lead to reliable deployment of automation in various applications like sensor networks, transportation networks, and manufacturing. This may potentially lead to the replacement of repetitive tasks or jobs that are traditionally performed by humans. However, automation and artificial intelligence (AI) can lead to better productivity, efficiency, and cost-effectiveness with an overall increase in societal living standards. By incorporating the ability to defend against adversarial attacks, our research can lead to more secure and robust adoption of AI technologies. However, potential failures and engineering issues remain challenging open problems.