Geometric Distillation for Graph Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

We study a new paradigm of knowledge transfer in the context of geometric deep learning, which aims at distilling knowledge from a teacher graph neural network (GNN) model trained on a large graph to a student GNN model operating on a smaller graph. To this end, we revisit the connection between thermodynamics and the behavior of GNN, based on which we propose Neural Heat Kernel (NHK) to encapsulate the geometric property of the underlying manifold concerning the architecture of GNN. A natural solution is derived by analysing and aligning NHKs on teacher and student models, dubbed as Geometric Knowledge Distillation. We develop non- and parametric instantiations and demonstrate their efficacy in various experimental settings for knowledge distillation regarding different types of privileged topological information and teacher-student schemes.

1 Introduction

Modern graph neural networks (GNNs) [26, 46, 50] have shown remarkable performance in learning representations for structured instances. From the perspective of geometric deep learning [5, 4, 35], much of the achievement of GNNs can be attributed to their successful implementation of the permutation invariance property as *geometric priors* [1] into the architecture design. Nevertheless, in practice, GNNs highly rely on graph topology, as essential input information, to explore the relational knowledge implicit in interactions of instance pairs throughout the entire message passing process, termed as *geometric knowledge* in this paper. As advances in generalized distillation [30, 44] reveal the possibility of encoding input features into model construction, natural questions arise as to:

Is it possible, and if so, how can we encode graph topology as a special type of 'geometric prior' into a GNN model, such that the model could precisely capture the underlying geometric knowledge even without full graph topology as input?.

In specific, we are interested in the following *geometric knowledge transfer* problem: a GNN model (with node-specific outputs for node-level prediction [22]) is exposed with a partial graph, which is a subset of the complete graph. Formally speaking, we have notations:

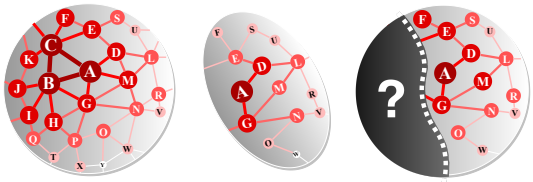
$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\} \text{ (partial graph)}, \ \tilde{\mathcal{G}} = \{\tilde{\mathcal{V}}, \tilde{\mathcal{E}}\} \text{ (complete graph)}, \ \text{where } \mathcal{V} \subseteq \tilde{\mathcal{V}}, \mathcal{E} \subseteq \{\mathcal{V} \times \mathcal{V}\} \cap \tilde{\mathcal{E}}. \quad (1)$$

Our goal is to transfer or encode geometric knowledge extracted from $\tilde{\mathcal{G}}$ to the target GNN model that is only aware of $\mathcal{G}$. Besides of academic interest, studying this problem is also of much practical value. As a non-exhaustive list of applications: improving efficiency without compromising on effectiveness for coarsened graphs [14, 23, 60], privacy constrained scenarios in social recommenders or federated learning where the complete graph is unavailable [31, 47, 59], promoting concentration on targeted community to bring up economic benefits [51].

Achieving this target is non-trivial for that we need to first find a principled and fundamental way to encapsulate the geometric knowledge extracted by GNN model, which requires in-depth investigation

[1] Geometric priors originally refer to the geometric principles naturally encoded in deep learning architectures, e.g., translational symmetry for CNNs, permutation invariance for GNNs.

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



a) Teacher b) Student (before GKD) c) Student (after GKD)

Figure 1: Feature propagation on the underlying manifold $\mathcal{M}$. (a) Teacher: aware of the complete graph topology, and faithfully explore geometric knowledge about the underlying manifold. (b) Student before GKD: only aware of partial graph topology, and estimate biased geometry property. (c) Student after GKD: able to propagate features on the same space as teacher by alignment of NHKs.

on the role of graph topology throughout the progressive process of message passing. Therefore, we take a thermodynamic view borrowed from physics and propose a new methodology built upon recent advances revealing the connection between heat diffusion and architectures of GNNs [8, 48, 7]. Specifically, we interpret feature propagation as heat flows on the underlying Riemmanian manifold, whose characteristics (that are dependent on graph topology and the GNN model) pave the way for a principled representation of the latent geometric knowledge.

1.1 Our Contributions

New theoretical perspective for analyzing latent graph geometry. On top of the connection between heat equation and GNNs, we step further to inspect the implication of heat kernel for GNNs, and propose a novel notion of *Neural Heat Kernel* (NHK) with rigorous proof of its existence. Heat kernel intrinsically defines the unique solution to the heat equation and can be a fundamental characterization for the geometric property of the underlying manifold [17, 18]. Likewise, NHK uncovers geometric property of the latent graph manifold for GNNs, and governs how information flows between pairs of instances, which lends us a mathematical tool to encapsulate geometric knowledge extracted from GNN model and enables geometric knowledge transfer. We also highlight this result alone may be of interest in the field of geometric deep learning, and is valuable for understanding GNNs.

Flexible distillation framework with versatile instantiations. Based on the insights mentioned above, we treat NHK matrices as representation of the latent geometric knowledge, upon which we build a flexible and principled distillation framework dubbed as *Geometric Knowledge Distillation* (GKD), which aims at encoding and transferring geometric knowledge by aligning latent manifolds behind GNN models as illustrated in Fig. 1. Moreover, we develop non-parametirc and parametric versions of GKD, in terms of different ways to approximate NHK computation. Specifically, the former derives explicit NHKs via assumptions on latent space, and the later learns NHK in a data-driven manner through the lens of variational inference, which will be shown to match two models by minimizing the KL-divergence of their distributions.

Promising results for geometric knowledge transfer and conventional KD purposes. We verify the efficacy of GKD in terms of different geometric knowledge types (i.e., edge-aware and node-aware ones), and further show its effectiveness for conventional KD purposes (e.g., model compression, self-distillation) for broader applicability. We highlight that our methods consistently exceed teacher model and rival with the oracle model that gives the performance upper bound in principle.

1.2 Links to Related Works

Geometric Deep Learning. The study of geometric deep learning [5, 35] provides fundamental principles and methodology to generalize deep learning methods to non-Euclidean domains (e.g., graphs and manifolds). From this perspective, architectures for off-the-shelf GNNs [26, 46, 50, 19] have naturally incorporated the geometric prior knowledge for graphs such as *permutation invariance*. Despite their remarkable success, they highly rely on the graph topology. This work extends the idea of geometric deep learning by treating the global graph topology as a special type of prior knowledge, and attempts to encode it into GNNs themselves, such that the trained model would leverage information from the global graph topology even without explicitly taking it as input.

Graph-Based Knowledge Distillation. Knowledge distillation (KD) [21, 1] uses the outputs of a teacher model as alternative supervised signals to teach a student model, with various new paradigms including feature-based [38, 58] and relation-based [56] ones. While some prior arts [9, 54, 47, 55, 53] attempted to combine KD and GNNs, i.e., graph-based KD, they are nearly straight-forward adaptations of KD without in-depth investigation on the role of graph topology, also restricted by

a specific choice of GNN architecture or application scenarios. In contrast, we first formalize the problem of geometric knowledge transfer, theoretically answer the question "how to represent graph geometric knowledge and encode it into GNN models", and propose geometric distillation approach based on the theoretical results. We also add introduction and comparison with other distilling methods to Appendix E.5.

2 Preliminaries

We commence with a brief detour to heat equation on Riemannian manifolds, and its connection with modern GNN architectures. Moreover, we bring forth the notion of *heat kernel* to motivate this work.

Heat Equation on Manifolds. We are interested in heat equation defined on a smooth k-dimensional Riemannian manifold $\mathcal{M}$. Suppose the manifold is associated with a scalar- or vector-valued function $x(u,t) : \mathcal{M} \times [0,\infty) \to \mathbb{R}^d$, quantifying a specific type of signals such as *heat* at a point $u \in \mathcal{M}$ and time t. Fourier's law of heat conductivity describes the flow of heat with respect to time and space, via a partial differential equation (PDE) called *heat equation* [6], i.e.,

$$\frac{\partial x(u,t)}{\partial t} = -c\,\Delta x(u,t), \tag{2}$$

where $c > 0$ is the *thermal conductivity* coefficient, and Δ is the natural *Laplace–Beltrami operator* associated with $\mathcal{M}$. Rewriting Δ as the functional composition of the *divergence operator* ∇^* and *gradient operator* ∇, i.e., $\Delta = \nabla^* \circ \nabla$, we can interpret the heat equation as: the variation of temperature within an infinitesimal time interval at a point is equivalent to the divergence between its own temperature and the average temperature on an infinitesimal sphere around it.

Implications on Graphs. A spatial discretisation of a continuous manifold yields a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, whose nodes can be thought of as embedded on the base manifold. In fact, the heat equation along with variants thereof (e.g., Schrödinger equation) have found widespread use in modeling graph dynamics [11, 24, 33]. More importantly, it has been recently revealed to be intimately related with the architectures of modern GNNs [48, 8, 7]: suppose $\mathbf{X}(0) = \{x(u,0)\}_{u\in\mathcal{V}} \in \mathbb{R}^{n\times d}$ denotes the initial condition for Eqn. (2) determined by input node features, then solving the heat equation under certain definitions of ∇^* and ∇ (i.e., definition of Δ) amounts to different architectures of GNNs. For instance:

Example 1. [48] Define the discretised counterpart of Δ as the graph Laplacian matrix $\mathbf{L} = \widetilde{\mathbf{D}}^{-\frac{1}{2}}(\widetilde{\mathbf{D}} - \widetilde{\mathbf{A}})\widetilde{\mathbf{D}}^{-\frac{1}{2}}$. Numerically solving Eqn. (2) using the forward Euler method with step size $\tau = 1$ yields the formulation of Simple Graph Convolution (SGC) [50], where Θ denote learnable transformation matrix

$$\hat{\mathbf{X}}(t) = \left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\right)^t \mathbf{X}(0), \quad \hat{\mathbf{Y}} = \mathrm{softmax}\left(\hat{\mathbf{X}}(t)\mathbf{\Theta}\right). \tag{3}$$

Example 2. [8] Define the gradient operator ∇_{ij} as the difference of source and target node features, the divergence operator ∇_i^* as the sum of features of all edges for the node. Numerically solving Eqn. (2) using the explicit Euler scheme with step size τ yields the following recursive formulation

$$\hat{\mathbf{X}}(t+\tau) = \tau\left(\mathbf{G} - I\right)\hat{\mathbf{X}}(t) + \hat{\mathbf{X}}(t) \tag{4}$$

where $\mathbf{G}$ is a diffusivity coefficient matrix in place of c.

Moreover, stacking a non-linear transformation layer after each step yields the formulation of Graph Convolution Networks (GCN) [26] for Eqn. (3), Graph Attention Networks (GAT) [46] with residual connection for Eqn. (4), and even more GNN architectures by virtue of the flexibility of interpretion for heat equation on graphs.

Heat Kernels Intriguingly, it turns out that the initial value problem of heat equation on any manifold $\mathcal{M}$ has a smallest positive fundamental solution depending on the Laplace operator Δ, known as the *heat kernel* [2]. It is denoted as a kernel function $\kappa(x,y,t)$, such that

$$x(u_i,t) = e^{-t\Delta}x(u_i,0) = \int_{\mathcal{M}} \kappa(u_i,u_j,t)x(u_j,0)\mathrm{d}\mu(u_j), \tag{5}$$

where μ is a non-negative measure associated with $\mathcal{M}$. In physics, the heat kernel $\kappa(x,y,t)$ can be interpreted as a transition density that describes the asymptotic behavior of a natural Brownian

motion on the manifold. Its formulation thus can be treated as *a unique reflection or representation of the geometry of the underlying manifold*. For example, if the manifold is a k-dimensional *Euclidean Space* $\mathbb{R}^k$ or a *Hyperbolic Space* $\mathbb{H}^k$, the explicit formula of heat kernel is respectively given by,

$$\kappa(u_i, u_j, t) = \frac{1}{(4\pi t)^{k/2}} \exp\left(-\frac{\rho^2}{4t}\right) \text{ and } \kappa(u_i, u_j, t) = \frac{(-1)^m}{2^m \pi^m} \frac{1}{(4\pi t)^{\frac{1}{2}}} \left(\frac{1}{\sinh \rho} \frac{\partial}{\partial \rho}\right)^m e^{-m^2 t - \frac{\rho^2}{4t}}, \quad (6)$$

where $\rho = d(u_i, u_j)$ denote geodesic distance. Heat kernel has also been adopted for graph-related applications such as community detection [29], graph clustering [52].

3 Extending Heat Kernel to GNNs

The starting point of this work is the development of *neural heat kernel*, built upon the previously-mentioned connection of GNNs and heat equation. As will be discussed later, this novel notion lends us a thermodynamic perspective to the intrinsic geometric property of the latent graph manifold embodied in GNNs, and hence paves the way for distilling geometric knowledge.

3.1 Neural Heat Kernel

Consider the graph signal $\mathbf{X}(t)$ at time t and node features $\mathbf{H}^{(l)}$ at layer l as interchangeable notions. Consequently, feature propagation using one layer of GNN amounts to heat diffusion on the base manifold $\mathcal{M}$ within a certain time interval τ, leading to the equivalences of $\mathbf{X}(t+\tau)$ and $\mathbf{H}^{(l+1)}$:

$$\mathbf{H}^{(l+1)} = f_\theta(\mathbf{H}^{(l)}, \mathcal{G}), \ \mathbf{X}(t+\tau) = e^{-\tau \Delta(f_\theta, \mathcal{G})} \mathbf{X}(t), \quad (7)$$

where f_θ denotes an arbitrary GNN model with parameter θ, and $\Delta(f_\theta, \mathcal{G})$ denotes a generalization of Laplace-Beltrami operator defined over the base manifold $\mathcal{M}$ associated with graph $\mathcal{G}$ and the arbitrary backbone GNN model f_θ.

In light of this connection, we consider a natural generalization of heat kernel for GNNs, termed as *neural heat kernel (NHK)* to highlight its difference with heat kernel in the thermodynamic context. In particular, a *single-layer* NHK is defined as a positive definite symmetric kernel function denoted as $\kappa_\theta^{(l)}(v_i, v_j)$, where the sub-script θ implies that it is associated with the architecture and parameters of the backbone GNN, and the super-script (l) implies that it is specific to each layer, analogous to the role of continuous time t in Eqn. (5).

Theorem 1. ***(Existence of Single-Layer NHK)*** *Suppose two expressions in Eqn. (7) are equivalent (see Appendix G for more discussions), then for any graph $\mathcal{G}$ and GNN model f_θ, there exist a unique single-layer NHK function $\kappa_\theta^{(l)}(\cdot)$ such that for any node $v_i \in \mathcal{V}$ and $l > 0$,*

$$\mathbf{h}_i^{(l)} = \sum_{v_j \in \mathcal{V}} \kappa_\theta^{(l)}(v_i, v_j) \cdot \mathbf{h}_j^{(l-1)} \mu(v_j) \quad (8)$$

where $\mathbf{h}_i^{(l)} \in \mathbb{R}^d$ denotes the feature of node v_i at l-th layer, and μ is a measure over vertices specified as the inverse of node degree $1/d_i$.

To push further, we can generalize NHK across multiple layers of GNN, termed as a *cross-layer NHK* $\kappa_\theta(v_i, v_j, l \mapsto l+k)$ (e.g., from l-th layer to $(l+k)$-th layer of GNN). Its existence could be induced recursively by the *semi-group identity property* of NHK concerning consecutive GNN layers.

Theorem 2. ***(Semigroup Identity Property of NHK)*** *The NHK satisfies the semigroup identity property: $\forall v_i, v_j \in \mathcal{V}$ and $l > 0$, there exists a cross-layer NHK across two consecutive layers*

$$\kappa_\theta(v_i, v_j, l \mapsto l+2) = \sum_{v_k \in \mathcal{V}} \kappa_\theta^{(l+1)}(v_i, v_k) \kappa_\theta^{(l+2)}(v_k, v_j) d\mu(v_k) \quad (9)$$

This theorem indicates that stacks of multiple GNN layers also constitute a valid kernel, i.e.,

$$\mathbf{h}_i^{(l+k)} = \sum_{v_j \in \mathcal{V}} \kappa_\theta(v_i, v_j, l \mapsto l+k) \cdot \mathbf{h}_j^{(l)} \mu(v_j). \quad (10)$$

Analogous to heat kernel as an unique characterization of the underlying space, NHK characterizes the geometric property of the latent graph manifold for GNNs. Additionally, NHK is dependent on GNN models through the definition of the associated Laplace-Beltrami operator $\Delta(f_\theta, \mathcal{G})$, inheriting the expressiveness of neural networks and varying through the course of training. Intuitively, NHK can be thought of as a model-driven encoding for topological information, encapsulating the geometric knowledge learned by GNNs into a tractable functional form.

3.2 Application in Geometric Distillation

Consider the problem of distilling geometric knowledge, which involves an intelligent teacher model f_{θ^*}, which is exposed to and pre-trained over the (relatively) *complete graph* $\tilde{\mathcal{G}} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$, and a student model f_θ that is exposed to the partial graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} \subseteq \tilde{\mathcal{V}}$ and $\mathcal{E} \subseteq \{\mathcal{V} \times \mathcal{V}\} \cap \tilde{\mathcal{E}}$. Our target is to train a student model (with the help of teacher model) that operates on $\mathcal{G}$ to be as competitive as models operating on $\tilde{\mathcal{G}}$ during inference. Since $\mathcal{G}$ is a sub-graph of $\tilde{\mathcal{G}}$, they should lie in the *same space* (i.e., latent manifold) governed by the underlying mechanism of data generation, and hence we expect student and teacher models to capture the *same geometric property* of this shared space. This leads to the principle of *Geometric Knowledge Distillation* (GKD): transfer the geometric knowledge of the intelligent teacher to the student such that the student can propagate features as if it is aware of the complete graph topology (see the example in Fig. 1).

To this end, we resort to *NHK matrices* on the teacher (resp. student) model over the complete (resp. partial) graph as instantiations of their geometric knowledge, denoted as

$$\text{(Teacher)} \quad \mathbf{K}_{\theta^*}(\tilde{\mathcal{G}}, l \mapsto l+k) = \{\kappa_{\theta^*}(v_i, v_j, l \mapsto l+k)\}_{|\tilde{\mathcal{V}}| \times |\tilde{\mathcal{V}}|},$$
$$\text{(Student)} \quad \mathbf{K}_{\theta}(\mathcal{G}, l \mapsto l+k) = \{\kappa_{\theta}(v_i, v_j, l \mapsto l+k)\}_{|\mathcal{V}| \times |\mathcal{V}|},$$

written compactly as $\mathbf{K}^{(l+1)}(\mathcal{G})$ when $k = 1$. The NHK matrix is a positive semi-definite symmetric matrix, and alike κ, is dependent on the GNN model f_θ and graph $\mathcal{G}$. Denote $\mathbf{K}^{(l)}_{\theta^*,\mathcal{V}}(\tilde{\mathcal{G}}) \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ as the sub-matrix of $\mathbf{K}^{(l)}_{\theta^*}(\tilde{\mathcal{G}})$ with row and column indices in $\mathcal{V}$. The distillation loss for GKD is

$$\mathcal{L}_{dis}(\mathbf{K}_{\theta^*,\mathcal{V}}, \mathbf{K}_\theta, l \mapsto l+k) = \mathrm{d}(\mathbf{K}_{\theta^*,\mathcal{V}}(\tilde{\mathcal{G}}, l \mapsto l+k), \mathbf{K}_\theta(\mathcal{G}, l \mapsto l+k)), \tag{11}$$

where $\mathrm{d}(\cdot, \cdot)$ is a similarity measure, for which we choose Frobenius distance as implementation, i.e.,

$$\mathrm{d}(\mathbf{K}_{\theta^*,\mathcal{V}}, \mathbf{K}_\theta) = \|(\mathbf{K}_{\theta^*,\mathcal{V}} - \mathbf{K}_\theta) \odot \mathbf{W}\|_{\mathrm{F}}^2, \quad \mathbf{W}_{v_i,v_j} = \begin{cases} 1 & \text{if} \quad (v_i, v_j) \in \mathcal{E} \\ \delta & \text{if} \quad (v_i, v_j) \notin \mathcal{E}. \end{cases} \tag{12}$$

where $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is a weighting matrix to trade-off distillation loss with respect to different node pairs depending on their connectivity. For $k = 1$, the loss can be re-written as $\mathcal{L}^{(l+1)}_{dis}(\mathbf{K}_{\theta^*,\mathcal{V}}, \mathbf{K}_\theta)$. Note that one can also specify different k for teacher and student models in Eqn. (11) in case when the teacher model is deeper. We also study the impact of different similarity measures in Appendix F.1.

4 Instantiations for Geometric Knowledge Distillation

Unfortunately, deriving explicit formulas for NHKs is prohibitively challenging due to introduction of non-linearity. To circumvent it, we propose two types of instantiations for GKD, i.e., non-parametric and parametric. The former considers explicit NHKs by making assumptions on the underlying space, and the latter resorts to variational inference techniques to learn NHK in a data-driven manner.

4.1 Non-Parametric Geometric Distillation

Deterministic Kernel. One instantiation of NHK is a *Gauss-Weierstrass kernel* in the form of Eqn. (6), by assuming the underlying space is a Euclidean space. Since the distillation loss in Eqn. (11) is a homogeneous function, we can remove its scaling factor and define NHK as

$$\text{(Gauss-Weierstrass NHK)} \quad \kappa_\theta(v_i, v_j, l \mapsto l+k) \triangleq \exp\left(-\frac{\|\mathbf{h}_i^{(l)} - \mathbf{h}_j^{(l)}\|_2^2}{4T}\right), \tag{13}$$

where T denotes the estimation of the accumulated time interval. Alternatively, we can use *Sigmoid kernel* and define non-parametric NHK as:

$$\text{(Sigmoid NHK)} \quad \kappa_\theta(v_i, v_j, l \mapsto l+k) \triangleq \tanh\left(a \, \langle \mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)} \rangle + b\right), \tag{14}$$

where a, b are positive constants depending on l and k. It is a natural and intuitive choice as similarity measurement and empirically found as-well effective, albeit does not correspond to any named manifold to our knowledge.

Randomized Kernel. We can also define *Randomized kernel* based on the following theorem.

Theorem 3. ***(Expansion of NHK)*** *Let $\{\varphi_{k'}\}_{k'=0}^{\infty}$ be orthonormal basis of eigenfunctions of $-\Delta(f_\theta, \mathcal{G})$ with eigenvalues $0 < \lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \ldots$, NHK allows the expansion:*

$$\kappa_\theta(v_i, v_j, l \mapsto l+k) = \sum_{k'=0}^{\infty} e^{-\lambda_{k'}T} \varphi_{k'}(v_i)^\top \varphi_{k'}(v_j). \tag{15}$$

Based on this result, we resort to the approximation of NHK by defining a randomized kernel in a similar form as Eqn. (15), leading to the following formulation of randomized NHK:

$$\text{(Randomized NHK)} \quad \kappa_\theta(v_i, v_j, l \mapsto l+k) \triangleq \frac{1}{m} \sum_{k'=0}^{m} e^{-\lambda_{k'}T} \left[\sigma\left(\boldsymbol{W}_{k'}\mathbf{h}_i\right)^\top \sigma\left(\boldsymbol{W}_{k'}\mathbf{h}_j\right)\right], \tag{16}$$

where $\sigma\left(\boldsymbol{W}_{k'}\mathbf{h}_i\right)$ is used to proximate $\varphi_{k'}\left(v_i\right)$, $\boldsymbol{W}_{k'} = \left[\boldsymbol{\phi}_{1,k'}, \boldsymbol{\phi}_{2,k'}, \cdots, \boldsymbol{\phi}_{s,k'}\right]^\top$ is a transformation matrix, $\boldsymbol{\phi} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{I}_d\right)$ is a d-dimensional random variable from Gaussian distribution. In fact, under certain choice of activation function σ, Eqn. (16) could approximate a diversity of kernels [37, 10]. This design essentially enforces the alignment of teacher and student for arbitrary underlying manifold.

Training Scheme. We follow the standard training paradigm in KD literature [21, 16]: the teacher is pre-trained by a supervised prediction loss involving all labeled nodes in $\tilde{\mathcal{V}}$. After teacher is well-trained, we fix θ^* and train the student model according to

$$\theta = \arg\min_\theta \mathcal{L}_{pre}(\hat{\mathbf{Y}}_\theta, \mathbf{Y}) + \frac{\alpha}{L} \sum_{l=1}^{L} \mathcal{L}_{dis}^{(l)}(\mathbf{K}_{\theta^*,\mathcal{V}}, \mathbf{K}_\theta), \tag{17}$$

where $\mathbf{Y}$ denotes ground-truth labels of labeled nodes in $\mathcal{V}$, and $\hat{\mathbf{Y}}_\theta$ denotes the predictions of student model f_θ on $\mathcal{G}$, $\mathcal{L}_{dis}$ is the distillation loss defined by Eqn. (11), L denotes the total number of layers.

4.2 Parametric Geometric Distillation

Inheriting the similar spirit of auto-encoding Bayes [25], we introduce a *variational inverse-NHK* that is independently parameterized, denoted as $\kappa_\phi^\dagger$, whose existence is guaranteed by the invertibility of NHK matrices. Together with κ_θ, they define a symmetric form characterizing feature propagation:

$$\text{(Forward)} \quad \mathbf{h}_i^{(l+k)} = \sum_{v_j \in \mathcal{V}} \kappa_\theta(v_i, v_j, l \mapsto l+k) \cdot \mathbf{h}_j^{(l)} \mu(v_j), \tag{18}$$

$$\text{(Backward)} \quad \mathbf{h}_i^{(l)} = \sum_{v_j \in \mathcal{V}} \kappa_\phi^\dagger(v_i, v_j, l+k \mapsto l) \cdot \mathbf{h}_j^{(l+k)} \mu(v_j). \tag{19}$$

In practice, we follow existing kernel learning approaches [49] and use deep architectures to parameterize the variational inverse-NHK as

$$\kappa_\phi^\dagger(v_i, v_j, l+k \mapsto l) = g_\phi(\mathbf{h}_i^{(l+k)})^\top g_\phi(\mathbf{h}_j^{(l+k)}), \tag{20}$$

where $g_\phi : \mathbb{R}^d \to \mathbb{R}^s$ is the associated learnable non-linear mapping. Given a pre-trained teacher model, distilling geometric knowledge boils down to 1) establishing equivalence of Eqn. (18) and Eqn. (19), and 2) matching pseudo-inverse NHK matrices for teacher and student models (respectively denoted as $\mathbf{K}_{\theta^*,\mathcal{V}}^\dagger$ and $\mathbf{K}_\theta^\dagger$ with clear meanings), leading to the training scheme as follows.

Training Scheme. Based on Eqn. (19), we can define a *reconstruction loss* with respect to the teacher model (similar applies to the student model) as

$$\mathcal{L}_{rec}(\mathbf{H}_t^{(l+k)}, \mathbf{H}_t^{(l)}) = \|\mathbf{K}_{\theta^*}^\dagger \tilde{\mathbf{D}}^{-1} \mathbf{H}_t^{(l+k)} - \mathbf{H}_t^{(l)}\|_F^2, \tag{21}$$

where $\tilde{\mathbf{D}}$ denotes the degree matrix over $\tilde{\mathbf{G}}$. Then, minimizing the reconstruction loss with fixed GNN model parameter θ amounts to optimizing the variational parameter ϕ, and minimizing prediction and distillation losses given fixed ϕ amounts to optimizing the student model parameter θ:

$$\phi \leftarrow \arg\min_\phi \quad \beta \mathcal{L}_{rec}\left(\mathbf{H}_t^{(l+k)}, \mathbf{H}_t^{(l)}\right) + (1-\beta)\mathcal{L}_{rec}\left(\mathbf{H}^{(l+k)}, \mathbf{H}^{(l)}\right), \tag{22}$$

$$\theta \leftarrow \arg\min_\theta \quad \mathcal{L}_{pre}\left(\hat{\mathbf{Y}}_\theta, \mathbf{Y}\right) + \alpha \mathcal{L}_{dis}\left(\mathbf{K}_{\theta^*,\mathcal{V}}^\dagger, \mathbf{K}_{\theta^*,\mathcal{V}}^\dagger, l+k \mapsto l\right). \tag{23}$$

Applying two steps iteratively adds up to an EM-like algorithm for training the student model. In practice, we set $l + k$ as the last layer, and l as the first layer to use as much information as possible.

Justification. We justify parametric GKD from a variational inference perspective. From Eqn. (18), the forward GNN model f_θ defines a model distribution $p_\theta(\mathbf{H}^{(l)}, \mathbf{H}^{(l+k)}, \mathbf{K}) = p_\theta(\mathbf{H}^{(l)}) p_\theta(\mathbf{K}|\mathbf{H}^{(l)}) p_\theta(\mathbf{H}^{(l+k)}|\mathbf{K}, \mathbf{H}^{(l)})$, where $p_\theta(\mathbf{K}|\mathbf{H}^{(l)})$ is intractable, hindering the proceeding distillation. In this light, the variational inverse-NHK model $\kappa_\phi^\dagger$ is proposed with a variational distribution $q_\phi(\mathbf{H}^{(l)}, \mathbf{H}^{(l+k)}, \mathbf{K}) = q_\phi(\mathbf{H}^{(l+k)}) q_\phi(\mathbf{K}|\mathbf{H}^{(l+k)}) q_\phi(\mathbf{H}^{(l)}|\mathbf{H}^{(l+k)}, \mathbf{K})$, which has a tractable posterior $q_\phi(\mathbf{K}|\mathbf{H}^{(l+k)})$. Now, we justify our training scheme with iterative optimization for Eqn. (22) and (23) by the following proposition and refer the reader to Appendix D for detailed reasoning.

Proposition 1. *The optimization in Eqn.* (22) *and* (23) *essentially minimizes the following Kullback–Leibler (KL) divergence,*

$$\min_{\theta,\phi} \mathcal{D}_{kl} \left(q_\phi(\mathbf{K}, \mathbf{H}^{(l)}, \mathbf{H}^{(l+k)}) \parallel p_\theta(\mathbf{K}, \mathbf{H}^{(l)}, \mathbf{H}^{(l+k)}) \right), \tag{24}$$

and hence attempts to establish equivalence between two latent variable models p_θ and q_ϕ.

5 Experiments

We conduct experiments to validate the efficacy of our method on graph-structured data in terms of various types of privileged geometric knowledge, combinations of teacher-student GNN architectures and potential application scenarios. We use three benchmark datasets Cora [32], Citeseer [40], Pubmed [36], and two larger datasets, i.e., OGBN-Proteins and OGBN-Arxiv [22], for node classification tasks. More implementation details, experimental results, and discussions on scalability are respectively given in Appendix E, F, and H.

Implementation and Competitors We consider the following variants of the proposed GKD. 1) *GKD-G*: non-parametric Gaussian NHK; 2) *GKD-S*: non-parametric Sigmoid NHK; 3) *GKD-R*: randomized NHK; 3) *VGKD*: parametric variational NHK. We choose KD methods that is representative in its own category for comparison, including *KD* [21], *FitNets* [38],*FSP* [56],*TinyGNN* [53],*LSP* [55]. Implementation details and descriptions of baselines are deferred to Appendix. We also report the performances of teacher and student model trained with the standard classification loss, short as *Teacher* and *Student*. The teacher model is trained using the complete graph $\tilde{\mathcal{G}}$, and, to calibrate with all other methods, tested using the partial graph $\mathcal{G}$. Besides, we consider an *Oracle* model which is both trained and tested on $\tilde{\mathcal{G}}$, which naturally takes an advantaged place given more information during inference. Since our method is compatible with the vanilla KD paradigm [21], we report the performance delivered by their combinations (i.e., GKD+KD and VGKD+KD).

Experiment Settings We investigate on various experimental settings according to different types of privileged geometric knowledge. In the case of *edge-aware geometric knowledge*, the teacher model has access to additional edge information, i.e., $\mathcal{E} \subset \tilde{\mathcal{E}}$ and $\mathcal{V} = \tilde{\mathcal{V}}$. In the case of *node-aware geometric knowledge*, the teacher model has access to additional node information, i.e., $\mathcal{V} \subset \tilde{\mathcal{V}}$ and $\mathcal{E} = \tilde{\mathcal{E}} \cap \{\mathcal{V} \times \mathcal{V}\}$. We also consider other conventional or novel settings including model compression, self-distillation, teaching MLP, which will be illustrated in detail. The backbone f_θ is set as 3-layer GCN [26] for both student and teacher models, unless otherwise stated. All the experiments are repeated five times with random initialization.

5.1 Edge-Aware Geometric Knowledge

We report results for the edge-aware geometric knowledge setting, where the teacher model has access to auxiliary edges. To quantify the privileged information, we set the quantity $(|\tilde{\mathcal{E}}| - |\mathcal{E}|)/|\tilde{\mathcal{E}}|$, called *privileged information ratio* (PIR), as 0.5. As shown in Tab. 1, all variants of GKD outperform other KD baselines by a large margin on both datasets, and significantly exceeds both Student and Teacher models. Further, GKD and its variants rival, if not surpass, the Oracle model. In other words, the student model trained using GKD could use far less graph topological information to achieve very close performance to competitors that are aware of the full graph topology during inference. This implies that our GKD framework could indeed effectively transfer topological information.

With regard to comparison among the variants of GKD, the parametric VGKD performs better than its non-parametric counterparts in most cases, and GKD-R is the most effective non-parametric method

	Cora	CiteSeer	PubMed
Oracle	88.63 ± 0.48	73.64 ± 0.48	87.16 ± 0.19
Teacher	84.61 ± 0.37	70.88 ± 0.62	84.42 ± 0.52
Student	83.84 ± 1.32	69.94 ± 0.76	85.35 ± 0.43
KD	84.84 ± 1.19	70.04 ± 0.37	85.58 ± 0.32
FitNets	83.72 ± 1.45	69.99 ± 0.56	85.66 ± 0.27
FSP	83.55 ± 2.19	71.43 ± 1.26	85.46 ± 0.34
TinyGNN	83.92 ± 0.53	71.62 ± 0.31	83.92 ± 0.53
LSP	83.99 ± 1.39	70.23 ± 0.79	85.37 ± 0.49
GKD-G	87.68 ± 1.07	73.04 ± 0.70	85.74 ± 0.38
GKD-S	88.01 ± 0.79	72.46 ± 0.52	85.94 ± 0.43
GKD-R	$\mathbf{88.48 \pm 0.59}$	72.97 ± 0.53	86.19 ± 0.55
VGKD	88.41 ± 0.62	$\mathbf{73.12 \pm 0.58}$	$\mathbf{86.41 \pm 0.24}$
GKD+KD	88.95 ± 0.30	73.21 ± 0.53	86.29 ± 0.28
VGKD+KD	$\mathbf{89.09 \pm 0.40}$	$\mathbf{73.45 \pm 0.48}$	$\mathbf{86.48 \pm 0.52}$

Table 1: Results of node classification accuracy for the edge-aware knowledge setting.

	Cora	CiteSeer	PubMed
Oracle	88.63 ± 0.48	73.64 ± 0.48	87.16 ± 0.19
Teacher	87.27 ± 0.51	72.92 ± 0.90	85.98 ± 0.23
Student	84.84 ± 1.61	70.32 ± 1.12	84.74 ± 0.27
KD	86.71 ± 0.77	71.96 ± 1.10	85.55 ± 0.45
FitNets	86.09 ± 1.12	72.00 ± 0.78	85.78 ± 0.26
FSP	85.85 ± 1.66	70.92 ± 1.46	85.20 ± 0.45
TinyGNN	86.13 ± 0.52	70.84 ± 0.73	84.12 ± 0.51
LSP	85.67 ± 1.22	70.66 ± 1.01	85.71 ± 0.50
GKD-G	88.66 ± 0.85	73.18 ± 0.88	86.07 ± 0.45
GKD-S	88.54 ± 0.52	72.85 ± 0.57	86.10 ± 0.42
GKD-R	88.98 ± 0.39	72.80 ± 0.22	$\mathbf{86.16 \pm 0.33}$
VGKD	$\mathbf{89.15 \pm 0.45}$	$\mathbf{73.33 \pm 0.36}$	86.09 ± 0.54
GKD+KD	89.10 ± 0.44	72.94 ± 0.80	$\mathbf{86.24 \pm 0.26}$
VGKD+KD	$\mathbf{89.23 \pm 0.61}$	$\mathbf{73.41 \pm 0.60}$	86.20 ± 0.36

Table 2: Results of node classification accuracy for the node-aware setting.

Dataset	Setting	Oracle	Teacher	Student	KD	GKD	VGKD
OGBN-Proteins	Edge-Aware	72.39 ± 0.52	70.89 ± 0.43	67.75 ± 0.21	69.51 ± 0.36	71.84 ± 0.32	$\mathbf{72.18 \pm 0.34}$
	Node-Aware	72.39 ± 0.52	71.22 ± 0.62	68.90 ± 0.42	70.23 ± 0.32	71.92 ± 0.25	$\mathbf{71.99 \pm 0.44}$
OGBN-Arxiv	Edge-Aware	71.46 ± 0.41	67.96 ± 0.78	66.41 ± 0.45	68.63 ± 1.21	70.90 ± 0.80	$\mathbf{71.38 \pm 1.01}$
	Node-Aware	71.46 ± 0.41	69.35 ± 0.72	67.49 ± 0.65	68.86 ± 0.66	$\mathbf{71.31 \pm 0.83}$	71.27 ± 0.70

Table 3: Results of ROC-AUC on OGBN-Proteins dataset, and accuracy on OGBN-Arxiv dataset.

in general. Despite these results, GKD-G and GKD-S are also effective while being simpler and more easy-to-implement. We presume that the performance variation of different GKD realizations stem from the different geometric property governed by the feature of datasets. Thus, it is useful to choose a proper implementation of GKD that is suitable for the application.

5.2 Node-Aware Geometric Knowledge

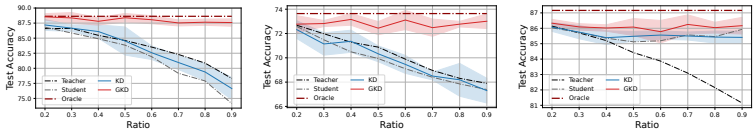
We further investigate on the node-aware geometric knowledge setting where the teacher model has access to more labeled nodes and their relations with the rest nodes. We set the PIR w.r.t. labeled nodes, defined as $(|\tilde{\mathcal{V}}_{train}| - |\mathcal{V}_{train}|)/|\tilde{\mathcal{V}}_{train}|$, to 0.5. A unique challenge of this setting compared to the edge-ware counterpart is that, apart from graph topological information, the student model has less labeled training samples, putting it into disadvantage. As we can see from Tab. 2. The proposed GKD and its variants again consistently outperform KD baselines throughout all the cases, surpasses both Student and Teacher models, and are even as competitive as Oracle.

5.3 Results on Larger Datasets

Table 3 presents results on two large graphs, i.e., OGBN-Proteins and OGBN-Arxiv. We follow the data split in original paper [22], and use the same PIR setting as citation networks, and choose the best variant of GKD to report in the table. As shown in the table, again, we found our methods consistently outperform Teacher and Student models, and are close to the performance of the Oracle model, which suggests the effectiveness of GKD in large graphs. More details regarding how our methods scale to larger graphs are deferred to Appendix H.

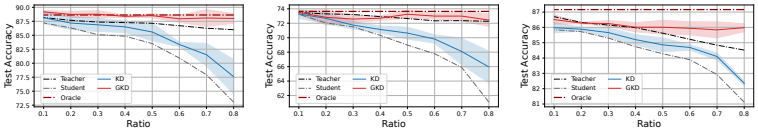
5.4 Performance Variation with Privileged Ratio

The results with respect to varying privileged (edge-aware and node-ware) information ratio are given respectively in Fig. 2 and Fig. 3. The performance of Oracle model is invariant as it is trained and tested on the same (complete) graph. In general, for Teacher model, Student model, and vanilla KD, their performance drops dramatically with increasing PIR quantifying the information loss, with vanilla KD's performance lie in-between. In contrast, our method is significantly more robust, only showing slight performance deterioration, exceeding the Teacher model, and approaching the Oracle model. Besides, we find an interesting phenomenon that in the edge-aware setting on Pubmed dataset, the performance of Teacher model is the worst. This is reasonable since the Teacher is trained using fully observed graph, and may perform poorly once the privileged part of graph topology, which it relies on for accurate results, becomes unavailable at test time.



(a) Cora (b) Citeseer (c) Pubmed

Figure 2: Performance variation with increasing PIR for the edge-aware knowledge setting.



(a) Cora (b) Citeseer (c) Pubmed

Figure 3: Performance variation with increasing PIR for the node-aware knowledge setting.

5.5 Other Settings

Setting	Teacher	Student	KD	GKD
Compression	GCN-64 88.76 ± 0.34	SGC 85.93 ± 0.17	SGC 86.32 ± 0.32	SGC 87.15 ± 0.85
		GCN-8 84.52 ± 1.34	GCN-8 87.50 ± 1.04	GCN-8 88.30 ± 0.46
		GCN-16 87.24 ± 0.43	GCN-16 88.09 ± 0.79	GCN-16 88.62 ± 0.50
Self-Distil	GCN-32 88.63 ± 0.48		GCN-32 88.98 ± 0.34	GCN-32 89.23 ± 0.52
	GCN-16 87.24 ± 0.43		GCN-16 87.62 ± 0.52	GCN-16 88.56 ± 0.40
Teach MLP	GCN-32 88.63 ± 0.48^1 76.75 ± 2.81^2	MLP-32 70.69 ± 0.57	MLP-32 72.61 ± 1.16	MLP-32 88.71 ± 0.58
	GCN-16 87.24 ± 0.43^1 73.21 ± 5.26^2	MLP-16 68.39 ± 0.87	MLP-16 71.91 ± 0.89	MLP-16 87.74 ± 0.32

[1] Using (resp., [2] Not using) graph at test time.

Table 4: Node classification accuracy on Cora in settings including: 1) model compression; 2) self-distillation; 3) teaching MLP.

Model Compression We report the performance of GKD in the conventional model compression setting [21] where the teacher and student are different-sized GNN models. In this setting, we train and test both teacher and student models on the complete graph. We use GCN as the backbone for teacher with a relatively large hidden size 64, and use SGC or GCN (with hidden size 8 and 16) as the student model. We report the best result among all variants of GKD. As shown in the first section of Tab. 4, our method achieves notable improvements over the student model, rendering it as a useful KD approach for model compression.

Self-Distillation We further report results for the setting of *self-distillation* [15, 34], which is a special case of KD when the teacher's and student's architectures are identical, often used for refining their own performance. The results as shown in Tab. 4 validate that GKD could also be used to effectively boost GNN's own performance.

GNNs Teach MLPs Finally, we study a special setting where any instance relation is unavailable at test time, i.e., $\mathcal{E} = \emptyset$, and examine whether our method could compress the geometric knowledge in this setting when GNN degrades to MLP. The results are shown in Tab. 4. Notably, the performance of conventional GNN model degrades dramatically when the topological information is not available at the test time, while our method turns out to still effective. It again verifies that our method could compress the geometric knowledge into the GNN, and also echoes recent advances in the domain of computer vision, which suggest that MLP could achieve competitive performance with CNN [42].

6 Conclusion

This paper formalizes the problem of graph topological knowledge transfer for GNNs. We investigate on the implication of heat kernel in GNNs and propose the novel notion of neural heat kernel. We leverage it to characterize the geometric property of the underlying manifold for graphs, and propose the framework of geometric knowledge distillation to transfer geometric knowledge from a teacher GNN model to a student GNN. Experimental results validate the effectiveness of our approach in various practical settings. The source code will be made public available. See limitations and potential social impact in Appendix I.

References

[1] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? *NeurIPS*, 27, 2014.

[2] Nicole Berline, Ezra Getzler, and Michele Vergne. *Heat kernels and Dirac operators*. Springer Science & Business Media, 2003.

[3] Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Liò, and Michael M Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *arXiv preprint arXiv:2202.04579*, 2022.

[4] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[5] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 2017.

[6] John Rozier Cannon. *The one-dimensional heat equation*. Number 23. Cambridge University Press, 1984.

[7] Benjamin Chamberlain, James Rowbottom, Davide Eynard, Francesco Di Giovanni, Xiaowen Dong, and Michael Bronstein. Beltrami flow and neural diffusion on graphs. *NeurIPS*, 2021.

[8] Benjamin Paul Chamberlain, James Rowbottom, Maria Gorinova, Stefan Webb, Emanuele Rossi, and Michael M Bronstein. Grand: Graph neural diffusion. *ICML*, 2021.

[9] Yuzhao Chen, Yatao Bian, Xi Xiao, Yu Rong, Tingyang Xu, and Junzhou Huang. On self-distilling graph neural network. *arXiv preprint arXiv:2011.02255*, 2020.

[10] Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. *NeurIPS*, 22:342–350, 2009.

[11] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[12] Francesco Di Giovanni, James Rowbottom, Benjamin P Chamberlain, Thomas Markovich, and Michael M Bronstein. Graph neural networks as gradient flows. *arXiv preprint arXiv:2206.10991*, 2022.

[13] Moshe Eliasof, Eldad Haber, and Eran Treister. Pde-gcn: Novel architectures for graph neural networks motivated by partial differential equations. *Advances in Neural Information Processing Systems*, 34:3836–3849, 2021.

[14] Matthew Fahrbach, Gramoz Goranci, Richard Peng, Sushant Sachdeva, and Chi Wang. Faster graph embeddings via coarsening. In *ICML*, 2020.

[15] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, pages 1607–1616. PMLR, 2018.

[16] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[17] Alexander Grigoryan. *Heat kernel and analysis on manifolds*, volume 47. American Mathematical Soc., 2009.

[18] Alexander Grigor'yan. Estimates of heat kernels on riemannian manifolds. *London Math. Soc. Lecture Note Ser*, 273:140–225, 1999.

[19] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pages 1025–1035, 2017.

[20] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *ICCV*, pages 1921–1930, 2019.

[21] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[22] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *NeurIPS*, 33:22118–22133, 2020.

[23] Yu Jin, Andreas Loukas, and Joseph JaJa. Graph coarsening with preserved spectral properties. In *AISTATS*, 2020.

[24] Matthias Keller and Daniel Lenz. Unbounded laplacians on graphs: basic spectral properties and the heat equation. *Mathematical Modelling of Natural Phenomena*, 5(4):198–224, 2010.

[25] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[26] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[27] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[28] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. *arXiv preprint arXiv:1911.05485*, 2019.

[29] Kyle Kloster and David F Gleich. Heat kernel based community detection. In *KDD*, pages 1386–1395, 2014.

[30] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *ICLR*, 2016.

[31] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. Recommender systems with social regularization. In *International Conference on Web Search and Web Data Mining*, pages 287–296, 2011.

[32] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 2000.

[33] Georgi S Medvedev. The nonlinear heat equation on dense graphs and graph limits. *SIAM Journal on Mathematical Analysis*, 46(4):2743–2766, 2014.

[34] Hossein Mobahi, Mehrdad Farajtabar, and Peter L Bartlett. Self-distillation amplifies regularization in hilbert space. *arXiv preprint arXiv:2002.05715*, 2020.

[35] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, pages 5115–5124, 2017.

[36] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and UMD EDU. Query-driven active surveying for collective classification. In *International Workshop on Mining and Learning with Graphs*, 2012.

[37] Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*.

[38] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[39] Laurent Saloff-Coste. *Aspects of Sobolev-type inequalities*, volume 289. Cambridge University Press, 2002.

[40] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 2008.

[41] Matthew Thorpe, Tan Minh Nguyen, Hedi Xia, Thomas Strohmer, Andrea Bertozzi, Stanley Osher, and Bao Wang. Grand++: Graph neural diffusion with a source term. In *International Conference on Learning Representations*, 2021.

[42] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

[43] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021.

[44] Vladimir Vapnik, Rauf Izmailov, et al. Learning using privileged information: similarity control and knowledge transfer. *JMLR*, 16(1):2023–2049, 2015.

[45] Dmitri V Vassilevich. Heat kernel expansion: user's manual. *Physics reports*, 388(5-6):279–360, 2003.

[46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *ICLR*, 2018.

[47] Shuai Wang, Kun Zhang, Le Wu, Haiping Ma, Richang Hong, and Meng Wang. Privileged graph distillation for cold start recommendation. *arXiv preprint arXiv:2105.14975*, 2021.

[48] Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. Dissecting the diffusion process in linear graph convolutional networks. *NeurIPS*, 2021.

[49] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

[50] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *ICML*, pages 6861–6871, 2019.

[51] Shu Wu, Mengqi Zhang, Xin Jiang, Xu Ke, and Liang Wang. Personalizing graph neural networks with attention mechanism for session-based recommendation. *TKDE*, 2019.

[52] Bai Xiao, Edwin R Hancock, and Richard C Wilson. Geometric characterization and clustering of graphs using heat kernel embeddings. *Image and Vision Computing*, 28(6):1003–1021, 2010.

[53] Bencheng Yan, Chaokun Wang, Gaoyang Guo, and Yunkai Lou. Tinygnn: Learning efficient graph neural networks. In *KDD*, pages 1848–1856, 2020.

[54] Cheng Yang, Jiawei Liu, and Chuan Shi. Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework. In *WWW*, pages 1227–1237, 2021.

[55] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. Distilling knowledge from graph convolutional networks. In *CVPR*, 2020.

[56] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, 2017.

[57] Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1285–1294, 2017.

[58] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *ICLR*, 2017.

[59] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. Subgraph federated learning with missing neighbor generation. *NeurIPS*, 34, 2021.

[60] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. Graph-less neural networks: Teaching old mlps new tricks via distillation. *arXiv preprint arXiv:2110.08727*, 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]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (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]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A,B,C,D
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 Appendix E for dataset information. See the supplementary for our codes
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix E,F
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See the experiment section
   (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 Appendix E
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 Appendix E
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
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]

A Proof for Theorem 1

Suppose $(\mathcal{M}, \mu)$ is the base manifold of dimension n with respect to graph $\mathcal{G}$ and GNN model f_θ, and $\Delta_\mu(\mathcal{G}, f_\theta)$ is the associated weighted Laplace operator with respect to $\mathcal{G}$ and f_θ (in the following shortened as Δ_μ). We leverage a Sobolev inequality on manifolds as Lemma 1.

Lemma 1. *[39] Let x be a function from the local Sobolev space $\mathcal{W}^{2\sigma}_{loc}(\mathcal{M})$ for a positive integer $\sigma > n/4$. Then, for any relatively compact open set $\Omega \subset \mathcal{M}$ and any set $K \Subset \Omega$, there is a constant C such that*

$$\sup_K |x| \leq C\|x\|_{\mathcal{W}^{2\sigma}(\Omega)}. \tag{25}$$

where the norm $\|\cdot\|_{\mathcal{W}^{2\sigma}}$ is defined as

$$\|x\|^2_{\mathcal{W}^{2\sigma}} = \sum_{l=0}^{k} \left\|\Delta^l_\mu x\right\|^2_{L^2} \tag{26}$$

Suppose σ is the smallest integer larger than $n/4$, and $\mathbf{P}_t = e^{-t\mathcal{L}}$ is the heat kernel semigroup, where $\mathcal{L} = -\Delta_\mu|_{W^2_0}$ is the Dirichlet Laplace operator for the base manifold regarding graph $\mathcal{G}$ and GNN model f_θ, we have the following lemma.

Lemma 2. *For any function $x \in L^2(\mathcal{M})$, $t > 0$, and set $K \Subset \mathcal{M}$, it holds that*

$$\sup_K |\mathbf{P}_t x| \leq C\left(1 + t^{-\sigma}\right)\|x\|_{L^2(\mathcal{M})}, \tag{27}$$

where C is a constant depending on $K, \mathbf{g}, \mu, n$.

Proof. Suppose $\{E_\lambda\}$ is the spectral resolution of the Dirichlet operator $\mathcal{L}$ for the base manifold. Consider the function $\Phi(\lambda) = \lambda^k e^{-t\lambda}$, where $t > 0$ and $k \in \mathbb{Z}^+$. We have

$$\mathcal{L}^k e^{-t\mathcal{L}} = \int_0^\infty \Phi(\lambda) dE_\lambda \tag{28}$$

Since $\Phi(\lambda)$ is bounded on $[0, +\infty)$, $\mathcal{L}^k e^{-t\mathcal{L}}$ is also bounded, and hence we have $\mathcal{L}^k\left(e^{-t\mathcal{L}}x\right) \in L^2(\mathcal{M})$. By noticing that the function $\lambda \mapsto \lambda^k e^{-t\lambda}$ takes its maximal value at $\lambda = k/t$, we have, for any $x \in L^2$,

$$\begin{aligned}
\left\|\Delta^k_\mu \mathbf{P}_t x\right\|_{L^2} &= \left\|\mathcal{L}^k e^{-t\mathcal{L}} x\right\|_{L^2} \\
&= \left(\int_0^\infty \left(\lambda^k e^{-t\lambda}\right)^2 d\left\|E_\lambda x\right\|^2_{L^2}\right)^{1/2} \\
&\leq \sup_{\lambda \geq 0}\left(\lambda^k e^{-t\lambda}\right)\left(\int_0^\infty d\left\|E_\lambda x\right\|^2_{L^2}\right)^{1/2} \\
&= \left(\frac{k}{t}\right)^k e^{-k}\|x\|_{L^2}.
\end{aligned} \tag{29}$$

Using the definition of $\|\cdot\|_{\mathcal{W}^{2\sigma}}$ in Eqn. (26) and the result of Eqn. (29), we obtain

$$\begin{aligned}
\|\mathbf{P}_t x\|_{\mathcal{W}^{2\sigma}} =& \sum_{k=0}^{\sigma}\left(\frac{k}{t}\right)^k e^{-k}\|x\|_{L^2} \\
\leq& C\left(1 + \sum_{k=1}^{\sigma}\left(\frac{k}{t}\right)^k e^{-k}\right)\|x\|_{L^2} \\
\leq& C'\left(1 + t^{-\sigma}\right)\|x\|_{L^2}.
\end{aligned} \tag{30}$$

Substituting $\mathbf{P}_t x$ into Lemma. 1 and using the result of Eqn. 30 yields Eqn. 27, and thus completes the proof. □

Consider the following equivalent expressions describing heat diffusion and feature propagation respectively,

$$\mathbf{H}^{(l+1)} = f_\theta(\mathbf{H}^{(l)}, \mathcal{G}),\ \mathbf{X}(t+\tau) = e^{-\tau\Delta(f_\theta,\mathcal{G})}\mathbf{X}(t) \tag{31}$$

where τ is a constant dependent on f_θ and l. We can rewrite the single-layer NHK as a function p_{τ,v_i} such that for any $v_j \in \mathcal{V}$ (or equivalently $v_j \in \mathcal{M}$)

$$p_{\tau,v_i}(v_j) = \kappa_\theta^{(l)}(v_i, v_j). \tag{32}$$

We define $g : \mathcal{V} \to \mathbb{R}^d$ as a function that outputs the feature of a node in l-th layer, which is clearly Lebesgue integrable by thinking of nodes embedded on the manifold, i.e., $g \in L^2(\mathcal{M})$.

Proof for the existence of unique single-layer NHK boils down to proof that for any $v_i \in \mathcal{M}$, $l > 0$ (i.e., $\tau > 0$), there exists a unique function $p_{\tau,v_i} \in L^2(\mathcal{M})$ such that, for all g_θ,

$$\mathbf{P}_\tau g_\theta(v_i) = \int_{v_j} p_{\tau,v_i} g_\theta(v_j)\mathrm{d}v_j. \tag{33}$$

Fix a relatively compact set $K \Subset \mathcal{M}$. By Lemma 2, for all $\tau > 0$ and $g_\theta \in L^2(\mathcal{M})$, the function $\mathbf{P}_\tau g_\theta(v_i)$ admits the estimate

$$|\mathbf{P}_\tau g_\theta(v_i)| \le C\left(1+\tau^{-\sigma}\right)\|g_\theta(v_i)\|_{L^2(\mathcal{M})}. \tag{34}$$

Therefore, for fixed l and GNN model f_θ, the mapping $g_\theta \mapsto \mathbf{P}_\tau g_\theta$ is a bounded linear functional on $L^2(\mathcal{M})$. By the Riesz representation theorem, there exists a function $p_{\tau,v_i} \in L^2(\mathcal{M})$ such that

$$\mathbf{P}_\tau g_\theta = \left(p_{\tau,v_i}, g_\theta\right)_{L^2} \text{ for all } g_\theta \in L^2(\mathcal{M}), \tag{35}$$

where $(\cdot,\cdot)_{L^2}$ denotes inner product in L^2, whence Eqn. (33) follows. The uniqueness of p_{τ,v_i} is evident from Eqn. (33) since for any point $v_i \in \mathcal{M}$ there is a compact set K containing v_i, the function $p_{\tau,v}$ is defined for all $\tau > 0$ and $v \in \mathcal{M}$.

B Proof for Theorem 2

Lemma 3. *For all $v_i, v_j \in \mathcal{M}$ and $t > 0$, the inner product $(p_{s,x}, p_{t-s,y})$ does not depend on $s \in (0, t]$.*

Proof. Using $\mathbf{P}_{t+s} = \mathbf{P}_s\mathbf{P}_t$ (by the definition of $\mathbf{P}_t$), Eqn. (35), and the symmetry of $\mathbf{P}_t$, we obtain that for all $v \in \mathcal{M}, t, s > 0$, and $g_\theta \in L^2(\mathcal{M})$, it holds that

$$\begin{aligned}
\mathbf{P}_{t+s}g_\theta(v_i) &= \mathbf{P}_s\left(\mathbf{P}_t g_\theta\right)(v_i) \\
&= (\mathbf{P}_{s,v_i}, \mathbf{P}_t g_\theta) = (\mathbf{P}_t\mathbf{P}_{s,v_i}, g_\theta) \\
&= \int_{\mathcal{M}} \mathbf{P}_t\mathbf{P}_{s,v_i}(z) g_\theta(v_j)\mathrm{d}\mu(v_j) \\
&= \int_{\mathcal{M}} \left(\mathbf{P}_{t,v_j}, \mathbf{P}_{s,v_i}\right) g_\theta(v_j)\mathrm{d}\mu(v_j),
\end{aligned} \tag{36}$$

□

Lemma 3 implies that, for all $v_i, v_j \in \mathcal{M}$ and $0 < s \le t$

$$p_t(v_i, v_k) = \left(p_{s,v_i}, p_{t-s,v_k}\right). \tag{37}$$

Hence, it holds that

$$\begin{aligned}
&\int_{\mathcal{M}} p_t(v_i, v_j)p_s(v_j, v_k)d\mu(v_j) \\
&\qquad = (p_t(v_i,\cdot), p_s(v_k,\cdot)) = p_{t+s}(v_i, v_k).
\end{aligned} \tag{38}$$

By letting

$$s = \min\{\tau^{(l+1)}, \tau^{(l+2)}\}, \quad t = \max\{\tau^{(l+1)}, \tau^{(l+2)}\}, \tag{39}$$

we can rewrite Eqn. (38) as the expression of semigroup identity property of layer-wise NHK

$$\begin{aligned}
&\kappa_\theta(v_i, v_j, l \mapsto l+2) = \\
&\qquad \sum_{v_k \in \mathcal{V}} \kappa_\theta^{(l+1)}(v_i, v_k)\kappa_\theta^{(l+2)}(v_k, v_j)d\mu(v_k).
\end{aligned} \tag{40}$$

The proof also generalizes to the cross-layer case, inducing cross-layer NHK $\kappa_\theta(v_i, v_j, l \mapsto l+k)$.

C Proof for Theorem 3

Let $\{\varphi_k\}_{k=1}^{\infty}$ be an orthonormal basis of eigenfunctions of $\mathcal{L}$ with an increasing sequence of non-negative eigenvalues $\{\lambda_k\}_{k=1}^{\infty}$, where $\lambda_k \to +\infty$. In consideration of

$$(p_{\tau,v_i}, \varphi_k)_{L^2} = P_\tau \varphi_k(v_i) = e^{-\tau\mathcal{L}}\varphi_k(v_i) = e^{-\tau\lambda_k}\varphi_k(v_i), \tag{41}$$

we have the following expansion of p_{t,v_i} by referring to results in literature [45]

$$p_{t,v_i} = \sum_k e^{-t\lambda_k}\varphi_k(v_i)\varphi_k. \tag{42}$$

Let T be the accumulated time interval from l-th layer to $(l+k)$-th layer, in consideration of the equivalence shown in Eqn. (32), we could write Eqn. (42) as

$$\kappa_\theta(v_i, v_j, l \mapsto l+k) = \sum_{k'=0}^{\infty} e^{-\lambda_{k'}T}\varphi_{k'}(v_i)^\top \varphi_{k'}(v_j) \tag{43}$$

completing the proof.

D Proof for Proposition 1

In the following, we use $\mathbf{X}, \mathbf{Y}, \mathbf{K}$ to denote $\mathbf{H}^{(l)}$, $\mathbf{H}^{(l+k)}$, $\mathbf{K}(\mathcal{G}, l \mapsto l+k)$ for brevity. By definition in Section 4.2 we have a forward GNN model f_θ with the joint distribution of latent variables

$$p_\theta(\mathbf{X}, \mathbf{Y}, \mathbf{K}) = p_\theta(\mathbf{X})\ \underbrace{p_\theta(\mathbf{K}|\mathbf{X})}_{\text{Intractable}}\ p_\theta(\mathbf{Y}|\mathbf{X}, \mathbf{K}), \tag{44}$$

and a variational inverse-NHK model $\kappa_\phi^\dagger$ with the joint distribution of latent variables

$$q_\phi(\mathbf{X}, \mathbf{Y}, \mathbf{K}) = q_\phi(\mathbf{Y})\ \underbrace{q_\phi(\mathbf{K}|\mathbf{Y})}_{\text{Tractable}}\ q_\phi(\mathbf{X}|\mathbf{Y}, \mathbf{K}). \tag{45}$$

Based on these equations, we can write the KL-divergence between joint distributions of p_θ and q_ϕ as

$$\begin{aligned}
&\mathcal{D}_{\mathrm{KL}}(q_\phi(\mathbf{X}, \mathbf{Y}, \mathbf{K}) \parallel p_\theta(\mathbf{X}, \mathbf{Y}, \mathbf{K})) \\
&= \iiint q_\phi(\mathbf{X}, \mathbf{Y}, \mathbf{K}) \log \frac{q_\phi(\mathbf{X}, \mathbf{Y}, \mathbf{K})}{p_\theta(\mathbf{X}, \mathbf{Y}, \mathbf{K})} \mathrm{d}\mathbf{X}\mathrm{d}\mathbf{Y}\mathrm{d}\mathbf{K} \\
&= \mathbb{E}_{q_\phi(\mathbf{Y})}\left[\iint q_\phi(\mathbf{K}, \mathbf{X}|\mathbf{Y}) \log \frac{q_\phi(\mathbf{Y})q_\phi(\mathbf{K}, \mathbf{X}|\mathbf{Y})}{p_\theta(\mathbf{X}, \mathbf{Y}, \mathbf{K})} \mathrm{d}\mathbf{X}\mathrm{d}\mathbf{K}\right] \\
&= C' + \mathbb{E}_{q_\phi(\mathbf{Y})}\left[\iint q_\phi(\mathbf{K}, \mathbf{X}|\mathbf{Y}) \log \frac{q_\phi(\mathbf{K}, \mathbf{X}|\mathbf{Y})}{p_\theta(\mathbf{X}, \mathbf{Y}, \mathbf{K})} \mathrm{d}\mathbf{X}\mathrm{d}\mathbf{K}\right] \\
&= C' + \mathbb{E}_{q_\phi(\mathbf{Y})}\Big[\mathbb{E}_{q_\phi(\mathbf{K}|\mathbf{Y})}\Big[\int_{\mathbf{X}} q_\phi(\mathbf{X}|\mathbf{Y}, \mathbf{K}) \cdot \\
&\qquad\qquad \log \frac{q_\phi(\mathbf{K}|\mathbf{Y})q_\phi(\mathbf{X}|\mathbf{Y}, \mathbf{K})}{p_\theta(\mathbf{X}, \mathbf{Y}, \mathbf{K})} \mathrm{d}\mathbf{X}\Big]\Big] \\
&= C + \mathbb{E}_{q_\phi(\mathbf{Y})}\Big[\mathbb{E}_{q_\phi(\mathbf{K}|\mathbf{Y})}\Big[\int_{\mathbf{X}} q_\phi(\mathbf{X}|\mathbf{Y}, \mathbf{K}) \cdot \\
&\qquad\qquad \log \frac{q_\phi(\mathbf{X}|\mathbf{Y}, \mathbf{K})}{p_\theta(\mathbf{X})p_\theta(\mathbf{Y}, \mathbf{K}|\mathbf{X})} \mathrm{d}\mathbf{X}\Big]\Big] \\
&= C + \mathbb{E}_{q_\phi(\mathbf{Y})}\Big[\mathbb{E}_{q_\phi(\mathbf{K}|\mathbf{Y})}\Big[\underbrace{\mathcal{D}_{\mathrm{KL}}(q_\phi(\mathbf{X}|\mathbf{Y}, \mathbf{K}) \parallel p_\theta(\mathbf{X}))}_{\text{Reconstruction Loss}} - \underbrace{\mathbb{E}_{q_\phi(\mathbf{X}|\mathbf{Y}, \mathbf{K})}[\log p_\theta(\mathbf{K}, \mathbf{Y}|\mathbf{X})]}_{\text{Prediction Loss}}\Big]\Big].
\end{aligned} \tag{46}$$

The first term C is a constant entropy with respect to q_ϕ. The second term is the KL-divergence between the variational posterior $q_\phi(\mathbf{X}|\mathbf{Y}, \mathbf{K})$ and the prior $p_\theta(\mathbf{X})$, which corresponds to the reconstruction loss in Eqn. (22) that attempts to learn a NHK that faithfully reflects the latent heat diffusion process describing the GNN feature propagation. The third term is negative log-likelihood, which corresponds to the prediction loss in Eqn. (23) that attempts to fit the dataset. Therefore, minimizing this KL-divergence amounts to the iterative optimization scheme of Eqn. (22) and (23).

E Implementation Details

We present implementation details for our experiments for reproducibility. We implement our model as well as the baselines with Python 3.8, Pytorch 1.8 and Pytorch Geometric 2.0. All experiments are conducted on NVIDIA V100 with 16 GB memory. All parameters are initialized with Xavier initialization procedure. We train the model by Adam optimizer. Both teacher and student models are trained from scratch. For the main results reported in Tab. 1 and 2, we choose the backbone model as a 3-layer GCN with hidden size 32. In case that the graph is too large, we use mini-batch training (draw a mini-batch of nodes from the vertex set $\mathcal{V}$) for computing the distillation loss. We use the whole vertex set on Cora and Citeseer datasets, and choose the largest batch-size under the memory constraint for Pubmed dataset and other large datasets. More details can be found in Appendix H.

E.1 Dataset Description

We choose three benchmark citation network datasets, i.e., Cora, Citeseer and Pubmed, and two large-scale network datasets, i.e., OGB-Arxiv and OGB-Proteins for node classification. For citation networks, we randomly split them into train/valid/test data according to the ratio 2:1:1, in order to balance (i.e., 1:1:1:1) node number of each category (student/ privilege/ valid/ test) in the node-aware setting. For two large-scale datasets, we follow the original splitting [22] for evaluation. The statistics of these datasets are summarized in Tab. 5.

Dataset	# Classes	# Nodes	# Edges	Metric
Cora [32]	7	2,485	5,069	Accuracy
Citeseer [40]	6	2,120	3,679	Accuracy
PubMed [36]	3	19,717	44,324	Accuracy
OGB-Arxiv [22]	40	169,343	1,166,243	Accuracy
OGB-Proteins [22]	2	132,534	39,561,252	ROC-AUC

Table 5: Statistics of Datasets.

E.2 Implementation of non-parametric GKD

For Gauss-Weierstrass NHK in Eqn. (13), we treat the accumulated time interval T as a hyper-parameter that is consistent for all layers. For sigmoid NHK in Eqn. (13), we let $a = 1$, $b = 0$ and the NHK is simple dot-product with activation. For randomized NHK in Eqn. (16), we let $s = 2d$ for the random transformation matrix $\mathbf{W}$, $\lambda_{max} > 0$ and m are hyper-parameters, and we use $tanh(\cdot)$ as activation function for $\sigma(\cdot)$.

Algorithm 1: Training Algorithm for GKD.

Input: Complete graph $\tilde{\mathcal{G}}$ with labels $\tilde{\mathbf{Y}}$, partial graph $\mathcal{G}$ with labels $\mathbf{Y}$, learning rate γ, initial parameters θ^*, θ.

Training Teacher GNN:

while *Not converged* **do**

  Teacher model conducts feature propagation on $\tilde{\mathcal{G}}$

  Calculate $\mathcal{L}_{pre}(\hat{\mathbf{Y}}_{\theta^*}, \tilde{\mathbf{Y}})$ as in Eqn. (17)

  $\theta^* \leftarrow \theta^* - \gamma \nabla_{\theta^*} \mathcal{L}_{pre}(\hat{\mathbf{Y}}_{\theta}, \tilde{\mathbf{Y}})$

Save teacher model as f_{θ^*}

Training Student GNN:

Load teacher model f_{θ^*}

while *Not converged* **do**

  Teacher model conducts feature propagation on $\tilde{\mathcal{G}}$

  Student model conducts feature propagation on $\mathcal{G}$

  Calculate $\mathcal{L}_{pre}$ and $\mathcal{L}_{dis}$ as in Eqn. (17)

  $\theta \leftarrow \theta - \gamma \nabla_{\theta} \mathcal{L}_{pre}(\hat{\mathbf{Y}}_{\theta}, \mathbf{Y}) + \frac{\alpha}{L} \sum_{l=1}^{L} \mathcal{L}_{dis}^{(l)}$

E.3 Implementation of parametric GKD

For the case of parametric GKD, we realize the non-linear mapping g_ϕ as a neural network one-layer feed-forward neural network with $tanh(\cdot)$ activation, and set $s = 2d$, the same as the non-parametric setting. In case that the teacher model has larger hidden size than the student model, we follow standard approaches in feature-based knowledge distillation methods [20] and use independent mappings for teacher and student models that are customized for their own hidden sizes. We use an EM-style algorithm for training the student model as in Eqn. (22) and Eqn. (23). For variants of GKD+KD and VGKD+KD, we consider additional standard label-based distillation loss in [21] with respect to labeled nodes, inducing an additional hyper-parameter α_{kd} that controls its importance.

Algorithm 2: Training Algorithm for VGKD.

Input: Complete graph $\tilde{\mathcal{G}}$ with labels $\tilde{\mathbf{Y}}$, partial graph $\mathcal{G}$ with labels $\mathbf{Y}$, learning rate γ_1 and γ_2, initial parameters θ^*, θ, ϕ.

Training Teacher GNN:

while *Not converged* **do**

  Teacher model conducts feature propagation on $\tilde{\mathcal{G}}$

  Calculate $\mathcal{L}_{pre}(\hat{\mathbf{Y}}_{\theta^*}, \tilde{\mathbf{Y}})$ as in Eqn. (17)

  $\theta^* \leftarrow \theta^* - \gamma_1 \nabla_{\theta^*} \mathcal{L}_{pre}(\hat{\mathbf{Y}}_\theta, \tilde{\mathbf{Y}})$

Save teacher model as f_{θ^*}

Training Student GNN:

Load teacher model f_{θ^*}

while *Not converged* **do**

  Teacher model conducts feature propagation on $\tilde{\mathcal{G}}$

  Optimization for ϕ

  Student model conducts feature propagation on $\mathcal{G}$

  Calculate reconstruction loss $\mathcal{L}_{rec}$ as in Eqn. (22)

  $\phi \leftarrow \phi - \gamma_2 \nabla_\phi \mathcal{L}_{rec}$

  Optimization for θ

  Student model conducts feature propagation on $\mathcal{G}$

  Calculate $\mathcal{L}_{pre}$ and $\mathcal{L}_{dis}$ as in Eqn. (23)

  $\theta \leftarrow \theta - \gamma_1 \nabla_\theta \mathcal{L}_{pre}(\hat{\mathbf{Y}}_{\theta^*}, \mathbf{Y}) + \alpha \mathcal{L}_{dis}$

E.4 Descriptions and Implementations of Baselines

KD [21]: is the seminal work of knowledge distillation, which uses the predictions of the teacher model as soft labels to teach the student model.

FitNets [38]: is the representative work of feature-based knowledge distillation, using the intermediate layers of training instances to teach the student model.

FSP [56]: is a relation-based knowledge distillation method, using the Gram matrix between two intermediate layers to explore the relationships between different feature maps.

TinyGNN [53]: is a knowledge distillation method that adopts a neighbor distillation strategy to learn local structure knowledge from a deeper GNN implicitly.

LSP [55]: is a knowledge distillation method for graph convolutional networks, which aims to match the local distribution (similarity with adjacent nodes) of teacher and student models.

Note that some of these baselines (FitNets and FSP) are originally designed specifically for computer vision tasks, we adapt them to the setting of graph neural networks with slight modifications. While FSP is designed for graph neural networks, its original formulation of distillation loss does not account for the difference of graph topology for teacher and student models. To make it compatible with the settings considered in this paper, we fix the definition of local structures with respect to either student or teacher graph topology, and choose the best result for report. We refer to the hyper-parameter settings in their papers and also finetune them on different datasets.

E.5 Comparisons with Baselines

There are mainly four different types of distilling strategies, namely response-based KD [21] (which uses output layer of the teacher model to teach student), feature-based KD [38, 58, 20] (which matches intermediate layers of teacher and student), relation-based KD [56, 57] (which aligns the relationship between different layers or samples), and graph-based KD [53, 55] (which considers the graph information or designed for GNN). For experiments, we choose representative method from each category as baselines. While these existing distillation strategies have shown remarkable success in distillation tasks such as model compression, they rarely (carefully) study the role of graph geometry in GNN iterations. For geometric knowledge transfer task, it is crucial to find a fundamental and principled way to track how graph topology affects the behavior of a specific GNN. Therefore, we first probe the intersection between KD and geometric learning, propose NHK to characterize geometric knowledge and propose different variants of GKD that are shown to be effective especially in the geometric knowledge transfer setting.

F More Experimental Results

F.1 Different Similarity Measures

We further investigate on the impact of different similarity measures in the definition of geometric knowledge distillation loss function Eqn. (11). Specifically, we consider the following three similarity measures including, L2-Distance, L1-Distance and Dot-Product, whose corresponding matrix distance are respectively defined as:

$$\text{L2-Distance:} \quad \mathrm{d}(\mathbf{K}_{\theta^*,\mathcal{V}}, \mathbf{K}_\theta) = \|\mathbf{K}_{\theta^*,\mathcal{V}} - \mathbf{K}_\theta\|_{\mathrm{F}}^2, \tag{47}$$

$$\text{L1-Distance:} \quad \mathrm{d}(\mathbf{K}_{\theta^*,\mathcal{V}}, \mathbf{K}_\theta) = \mathrm{Sum}(|\mathbf{K}_{\theta^*,\mathcal{V}} - \mathbf{K}_\theta|), \tag{48}$$

$$\text{Dot-Product:} \quad \mathrm{d}(\mathbf{K}_{\theta^*,\mathcal{V}}, \mathbf{K}_\theta) = \mathrm{Sum}(\mathbf{K}_{\theta^*,\mathcal{V}} \odot \mathbf{K}_\theta), \tag{49}$$

The results are shown in Tab. 6,7 and 8. We can see that in most cases L2-Distance performs better than other similarity measurements, and thus we use its variant as formulation of our definition of matrix similarity measurement. Specifically, when $\delta = 1$, Eqn. (12) degrades to L2-Distance. Also, L1-Distance in some cases exceeds L2-Distance, which renders it as an alternative approach for similarity measurement in other scenarios. In contrast, Dot-Product only works in some rare cases and thus is out of our consideration in practice.

	L2-Distance	L1-Distance	Dot-Product
GKD-G	86.26 ± 1.34	87.39 ± 0.67	79.59 ± 1.84
GKD-S	87.09 ± 0.52	86.87 ± 0.58	61.03 ± 3.19
GKD-R	87.62 ± 0.46	87.53 ± 0.90	20.35 ± 1.77
VGKD	88.01 ± 0.62	87.81 ± 0.49	24.24 ± 1.02

Table 6: Ablation Study: Test (mean) accuracy and standard deviation for node classification results on Cora dataset.

	L2-Distance	L1-Distance	Dot-Product
GKD-G	71.50 ± 0.39	72.51 ± 0.77	68.24 ± 1.13
GKD-S	71.28 ± 1.35	71.74 ± 0.79	48.64 ± 2.62
GKD-R	72.20 ± 0.89	72.24 ± 0.94	26.24 ± 3.48
VGKD	72.89 ± 0.48	72.83 ± 0.23	25.87 ± 2.98

Table 7: Ablation Study: Test (mean) accuracy and standard deviation for node classification results on Citeseer dataset.

Notation	Description	Search Space
T	accumulated time interval	[0.25, 0.5, 1, 2, 4, 8]
m	the number of chosen functions	[1, 2, 4, 6, 8, 10, 15, 20]
λ_{max}	maximum value of eigenvalues	[0.5, 1, 2, 4, 8]
α	weight of the geometric distillation loss	[0.1, 0.3, 1.0, 3.0, 10.0, 30.0, 100.0, 300.0, 1000.0]
α_{kd}	weight of the label-based distillation loss	[0.0, 0.1, 0.2, 0.4, 0.6, 0.8]
τ	temperature for label-based distillation loss	[0.25, 0.5, 1, 2, 4, 10, 20]
δ	weight for non-connection entries for distillation loss	[0.0, 0.1, 0.2, 0.4, 0.6, 0.8, 1, 2]
β	weight teacher/student reconstruction loss	[0.0, 0.2, 0.4, 0.6, 0.8, 1]
γ	learning rate	[0.0001, 0.001, 0.01, 0.1]

Table 9: Parameter searching space for GKD and its variants.

α	0	0.3	1.0	3.0	10.0	30.0	100.0	300.0	1000.0
GKD-G	83.84	$83.87_{+0.03}$	$83.72_{-0.12}$	$84.17_{+0.33}$	$84.11_{+0.27}$	$84.82_{+0.98}$	$86.26_{+2.42}$	$\mathbf{87.44}_{+3.60}$	$87.12_{+3.28}$
GKD-S	83.84	$84.61_{+0.77}$	$85.97_{+2.13}$	$87.06_{+3.22}$	$87.30_{+3.46}$	$\mathbf{87.40}_{+3.56}$	$87.22_{+3.38}$	$87.10_{+3.26}$	86.85
GKD-R	83.84	$87.39_{+3.55}$	$\mathbf{87.53}_{+3.69}$	$87.30_{+3.46}$	$87.30_{+3.46}$	$87.06_{+3.22}$	$87.09_{+3.72}$	$87.12_{+3.75}$	$87.12_{+3.75}$
VGKD	83.84	$86.53_{+2.69}$	$87.12_{+3.28}$	$\mathbf{87.65}_{+3.81}$	$87.48_{+3.64}$	$87.38_{+3.54}$	$87.22_{+3.38}$	$87.10_{+3.26}$	$87.02_{+3.18}$

Table 10: Test (mean) accuracy for node classification results with respect to varying weight α on Cora dataset.

	L2-Distance	L1-Distance	Dot-Product
GKD-G	85.94 ± 0.46	85.58 ± 0.57	79.20 ± 0.71
GKD-S	85.29 ± 1.43	85.71 ± 0.84	46.25 ± 5.49
GKD-R	86.06 ± 0.57	85.81 ± 0.44	40.34 ± 4.07
VGKD	86.02 ± 0.43	85.93 ± 0.37	42.10 ± 5.17

Table 8: Ablation Study: Test (mean) accuracy and standard deviation for node classification results on Pubmed dataset.

F.2 Hyper-Parameter Analysis

For parameter tuning, we adopt grid search method to search for hyper-parameters on validation set. Since our model is only sensitive to α (which will be discussed next), one can use other more efficient searching strategies instead of grid search (e.g., coordinate descent, Bayesian optimization) to achieve very similar results. The descriptions for several hyper-parameters and their search spaces are shown in Tab. 9.

Specifically, we investigate on the impact of hyper-parameter α, namely the weight of the geometric distillation loss, on the model performance. The results are shown in Tab. 10. As we can see from the table, for all variants of GKD, the test accuracy increases and then falls with the hyper-parameter $alpha$ changing from low to high values, achieving the peak accuracy within $[1, 1000]$. Among all variants, VGKD has the best peak performance, while GKD-R is the most stable one whose performance does not vary dramatically with increasing α.

F.3 More Results on Social Networks

We supplement more experiment results on Reddit, a social network, as is shown in Table 11 in edge-aware setting. We found that our method still performs better than Teacher and close to Oracle.

	Oracle	Teacher	Student	KD	GKD-S	GKD-R
Edge-aware	89.3	81.8	73.6	80.5	85.7	86.8
Node-aware	89.3	80.6	70.2	79.4	87.4	87.9

Table 11: Tentative experimental results for test accuracy on social network dataset Reddit. Due to limited time, we will supplement standard deviation results and other baselines in the future version.

F.4 Evaluation for Online Distillation

Online distillation in dynamic graphs is also an interesting application scenario where our method might be useful. As an exploration in this direction, we provide new results for online distillation (where teacher and student are simultaneously trained) on Cora in Table 13. We can see that our model's performance is similar to that in the offline setting except for taking more time to converge (which is an intrinsic limitation of online distillation). As a pioneering in the direction of geometric knowledge distillation, we stay focused in our paper for a general methodology and standard evaluation settings in the experiments. We leave more investigation on online distillation, which is promising yet orthogonal to the main body of our work, for future works.

	Oracle	Teacher	Student	GKD-G	GKD-S	GKD-R	VGKD
Offline	88.63 ± 0.48	84.61 ± 0.37	83.84 ± 1.32	87.68 ± 1.07	88.01 ± 0.79	88.48 ± 0.59	88.41 ± 0.62
Online	88.63 ± 0.48	84.61 ± 0.37	83.84 ± 1.32	87.75 ± 0.65	87.63 ± 0.65	88.28 ± 0.80	88.50 ± 0.33

Table 12: Experimental results for online distillation on Cora in edge-aware setting.

	Oracle	Teacher	Student	GKD-G	GKD-S	GKD-R	VGKD
Offline	88.63 ± 0.48	87.27 ± 0.51	84.84 ± 1.61	88.66 ± 0.85	88.54 ± 0.52	88.98 ± 0.39	89.15 ± 0.45
Online	88.63 ± 0.48	87.27 ± 0.51	84.84 ± 1.61	88.32 ± 0.47	88.40 ± 0.80	89.00 ± 0.44	88.92 ± 0.53

Table 13: Experimental results for online distillation on Cora in node-aware setting.

G More Discussions on the Equivalence of Eqn.7

The equivalence of two equations in Eqn. 7 is based on recent works [8, 7, 48, 13, 12] that built connection between heat equation and GNN. The main result of these works is that by treating node features H as signal X (corresponding to $x(u,t)$ in heat equation Eqn. 2) on the graph, solving the heat equation with Euler scheme yields the formulation of a GNN layer. In other words, the GNN can be seen as the discretisations of the continuous diffusion process described by the heat equation. Correspondingly in Eqn. 2, the left equation is a general GNN layer (corresponding to discretized diffusion process), and the right equation is directly derived from Eqn. 2 (corresponding to continuous diffusion process).

Moreover, different definitions of Laplace-Beltrami operator Δ yield different GNNs. To be more specific, the simplest definition [8, 48] of Δ is by letting the gradient operator ∇ denote assigning each edge the difference of adjacent nodes' features, and the divergence operator ∇^* denote the summation of edge features obtained from last step. In this case, setting τ in Eqn. 2 as 1 yields SGC [50], and since it is being pointed out to be a potential cause of over-smoothing (which is one example to show the thermodynamic and geometric perspective is helpful for understanding GNNs), the authors in [48] further set smaller τ which yields DGC. Reinterpreting ∇^* as weighted sum of edge features yields GAT [8, 46] and any other GNN following the message passing scheme:

$$\text{MessagePassing}(\mathbf{h}_u, \mathcal{G}) = \sum_{v \in \mathcal{N}_u \cup u} s(\mathbf{h}_u, \mathbf{h}_v) \cdot \mathbf{h}_v, \tag{50}$$

where $\mathcal{N}_u$ denotes neighbored nodes of u, and s denotes a parametric or non-parametric similarity score. The neighborhood summation could also be modified by changing the definition of ∇, which yields GNNs based on learned structures (e.g., GDC [28]) and those with residual links (e.g., APPNP [27]). Further considering different discretization schemes (e.g., implicit scheme, multi-step schemes) yields different variants of GRAND [8, 7, 41] that are found more powerful than its simplification (presumably because it better matches the continuous diffusion process).

Unfortunately, not all GNNs have a simple form of Δ, and for some of them, whether there exists such Δ is an open research question. Therefore, we write the operator as $\Delta(f_\theta, \mathcal{G})$ to associate it with model f_θ and use equivalence in Eqn. 2 as an analytical assumption. The thermodynamic and geometric perspective used in the paper is fundamental and useful for studying the geometry property of GNNs, which has also been adopted by other works [3, 43, 48, 41, 8, 7]. For example,

[3] attempts to explain heterophily and oversmoothing problems in GNNs by connecting GNNs to the heat equation defined by the sheaf Laplace-Beltrami operator $\Delta_{\mathcal{F}}$ and propose new GNN architectures based on the theory. Similarly, [41, 8, 7] propose new GNN architectures / rewiring methods from the same thermodynamic perspective, [48] explains over-smoothing issue of SGC based on the same equivalence of GNN and solve of heat equation, and [43] explains the over-squashing issue from a geometric perspective by analyzing the curvature. A common trait of these works is to draw analogies between GNNs and differential geometry / diffusion process to obtain meaningful theoretical results (ours: NHK as a characterization of GNN's underlying geometry) that are used to guide implementation (ours: GKD for geometric knowledge transfer).

H Scalability Analysis

H.1 How Does Our Method Scale to Large Graphs?

Suppose the size of partial graph (i.e., the small one) is n, then the computation complexity for distillation loss (in both non-parametric and parametric cases) is $O(dn^2)$, and the extra space consumption is $O(n^2)$.

Directly using whole graph is impractical when the partial graph is large. Therefore, as mentioned in appendix E.2, we recommend using mini-batch training for large graphs, which has already been adopted for training two large-scale datasets in our experiment. It could also be seamlessly integrated with the original mini-batch method (by sampling ego-graphs) used by large graphs without further modification. Our method could be run in most GPUs within a reasonable amount of time. Moreover, we want to remark the following points:

- The distillation loss for mini-batch training is not biased as long as nodes are evenly sampled, and thus will not harm the performance. We support this by showing GKD's performance with decreasing batch size using mini-batch training in Table. 14.
- Notably, the distilled student model is efficient for inference because of sparser graph structure and perhaps smaller model size, which is an important factor for those who resort to KD.
- Even in the case where we somehow must use the whole large graph for calculating the loss (which is extremely rare or doesn't exist), there are ways to reduce the complexity (e.g., pre-computing teacher's NHK matrix, using low-rank approximation).

Batch Size	Whole graph	1024	512	256	128
GKD(-R)	88.48 ± 0.59	89.02 ± 0.52	88.48 ± 0.30	88.63 ± 1.07	88.28 ± 0.30
VGKD	88.41 ± 0.62	88.72 ± 0.53	88.82 ± 0.37	88.47 ± 0.31	88.18 ± 0.96

Table 14: Performance of GKD(-R) and VGKD using mini-batch training with decreasing batch size in edge-aware setting on Cora.

H.2 Convergence Speed

We also supplement more comparisons with other peer KD methods in terms of convergence epochs in Table 15. As we can see from the table:

- The EM-style algorithm used for training VGKD does not cost too much extra epochs for convergence than the non-parametric GKD. The result is intuitive by noting that the learnable mapping g_ϕ used in Eqn. (22) is an independent module since it only "calibrates" the distillation loss and does not affect GNN's feed-forward computation. It is worth mentioning that the model needs not to wait for Eqn. (22) to converge to achieve good performance (though this step is necessary for further improvement) considering the fact that even using random mapping function (i.e., GKD-R) is effective.
- Both proposed models can converge within reasonable number of epochs (less than 200). Such a result is important since we usually use a fixed epoch number (e.g. 200) as the default

setting for training as recommended in [26]. In this case, the training time is proportional to the time per epoch, which will be discussed in detail next.

- In broader context, existing KD methods [16, 21, 38, 56] naturally cost more time than the non-KD counterpart (the teacher and student models), which is even amplified in our geometric knowledge transfer setting because of the inconsistency of graph topology that makes it harder for student to mimic the behavior of teacher. Similarly, in our work, it is normal that the proposed GKD costs more time than the pure teacher/student models due to the additional loss term for transferring geometric knowledge. Such extra time is often acceptable and unimportant in this area due to that the more important factors of interest are the inference efficiency and effectiveness of student model.

We can see that the convergence speed of our models are on par with other KD baselines, while outperforming them significantly in terms of testing performance (as shown in Tab. 1,2,3).

	Teacher	Student	GKD(-S)	VGKD	KD	FitNets	FSP	LSP
Cora	21.6 ± 6.1	13.8 ± 3.7	46.2 ± 7.2	71.4 ± 10.1	38.4 ± 5.4	62.5 ± 10.7	94.2 ± 12.6	41.5 ± 7.4
Citeseer	17.0 ± 4.6	11.4 ± 6.0	33.4 ± 9.2	52.6 ± 11.2	29.0 ± 8.3	48.8 ± 12.9	72.6 ± 12.0	31.3 ± 5.4
Pubmed	90.2 ± 17.2	65.4 ± 14.8	61.6 ± 15.6	102.0 ± 12.5	60.6 ± 12.8	94.4 ± 20.6	122.4 ± 24.5	70.1 ± 14.0

Table 15: Comparison of convergence epochs (in five different runs) of different models.

H.3 Training Time

We report the the training time per epoch of different methods in Table 16.

Remarkably, our parametric version in the worst case takes less than two times of the training time of vanilla KD, and is even more efficient than the relation-based KD method FSP that aims to match flow of solution procedure (FSP) matrices. In practice, the whole training in citation networks (in one run) can be done within a minute, which is very efficient.

Also, our models successfully scale to large OGB datasets with comparable time as other models and yielding significantly better performance, which again verifies the efficacy of our models.

	Teacher	Student	GKD(-S)	VGKD	KD	FitNets	FSP	LSP
Cora (ms)	37.2	35.8	58.7	73.4	41.3	55.2	78.7	60.0
Citeseer (ms)	36.6	32.9	49.8	66.7	42.5	47.3	70.2	57.2
Pubmed (ms)	53.9	43.6	75.3	92.4	61.5	68.1	101.0	80.6
OGB-Arxiv (s)	2.43	2.30	2.91	3.38	2.62	2.75	3.64	3.06
OGB-Proteins (s)	4.46	4.25	6.05	7.90	4.89	5.35	8.41	6.22

Table 16: The averaged training time per epoch of different models.

I Limitations and Potential Impacts

We thoroughly discuss the limitations of the present work that we could identify at current stage from the following two aspects.

Theoretical Analysis. While the equivalence that connects GNNs and heat diffusion on manifold we use for theoretical analysis only strictly holds for some specific GNN instantiations including SGC, APPNP, GAT and more others, as illustrated in Appendix G, it is also adopted by many of other works as meaningful interpretations and insights [3, 48, 41, 8, 7]. In practice, this is fundamental and useful for studying the geometry property of GNNs and provides a principled perspective for understanding the behaviors of common GNN models based on which more complex GNN architectures (e.g., [41, 3]) are developed. Furthermore, the methodology introduced by this paper, i.e., the geometric knowledge distillation approaches with the non-parametric and parametric versions, is applicable for arbitrary off-the-shelf GNN architectures beyond the above-mentioned GNN instantiations. While our experiments mainly focus on GCN, SGC and GAT as basic representatives for evaluation purpose,

we believe the methods can potentially combine with more powerful models for further performance improvement in the problem of our interest and be extended to other promising applications involving geometric knowledge transfer, e.g., cross-graph domain generalization. We leave these interesting explorations for future works.

Time Cost and Complexity. The computation complexity of GKD is $O(dn^2)$ and the parametric instantiation may take some extra time to converge compared to the pure non-KD counterpart (pure teacher and student models), as specifically shown in Appendix H. However, such extra training time is acceptable and relatively unimportant given two facts: 1) the total training time on citation networks (in one run) is within one minute, which is quite efficient and convenient for model development; 2) with slight sacrifice on time cost, the advantage brought by our model is significant, i.e., allowing for geometric knowledge transfer between two models, which is non-trivial to achieve. Moreover, with mini-batch training, our model can successfully scale to large graphs, i.e., the OGB datasets, without compromising the performance on testing data, as shown in our experiments in Appendix H. There also exists other potential ways, such as low-rank approximation, as illustrated in Appendix H.1, which can be used for further accelerating the current models. As a pioneering work in geometric knowledge distillation, we stay focused on introduction of new methodology and leave such an orthogonal improvement for future investigation.

Finally, our work contributes from both theoretical and practical sides, and we do not foresee any direct negative societal impacts.