Node Embedding from Neural Hamiltonian Orbits in Graph Neural Networks

Qiyu Kang [*1] **Kai Zhao** [*1] **Yang Song** [2] **Sijie Wang** [1] **Wee Peng Tay** [1]

Abstract

In the graph node embedding problem, embedding spaces can vary significantly for different data types, leading to the need for different GNN model types. In this paper, we model the embedding update of a node feature as a Hamiltonian orbit over time. Since the Hamiltonian orbits generalize the exponential maps, this approach allows us to learn the underlying manifold of the graph in training, in contrast to most of the existing literature that assumes a fixed graph embedding manifold with a closed exponential map solution. Our proposed node embedding strategy can automatically learn, without extensive tuning, the underlying geometry of any given graph dataset even if it has diverse geometries. We test Hamiltonian functions of different forms and verify the performance of our approach on two graph node embedding downstream tasks: node classification and link prediction. Numerical experiments demonstrate that our approach adapts better to different types of graph datasets than popular state-of-the-art graph node embedding GNNs. The code is available at https://github.com/zknus/Hamiltonian-GNN.

1. Introduction

Graph neural networks (GNNs) (Yue et al., 2019; Ashoor et al., 2020; Kipf & Welling, 2017b; Zhang et al., 2022; Wu et al., 2021) have shown remarkable inference performance on graph-structured data, including, but not limited to, social media networks, citation networks, and molecular graphs in chemistry. Most existing GNNs embed graph nodes in Euclidean spaces without further consideration of the dataset graph geometry. For some graph structures like the tree-like graphs (Liu et al., 2019), the Euclidean space may not be a proper choice for the node embedding. Recently, hyperbolic GNNs (Chami et al., 2019; Liu et al., 2019) propose to embed nodes into a hyperbolic space instead of the conventional Euclidean space. It has been shown that tree-like graphs can be inferred more accurately by hyperbolic GNNs.

Real-world graphs often have complex and varied structures, which can be more effectively represented by utilizing different geometric spaces. As shown in Figure 1, the Gromov δ-hyperbolicity distributions[1] (Gromov, 1987) of various datasets exhibit a range of diverse values. This indicates that it is not optimal to embed each dataset with diverse geometries into a single globally homogeneous geometry. Works like (Zhu et al., 2020b) have attempted to embed graph nodes in a mixture of the Euclidean and hyperbolic spaces, where the intrinsic graph local geometry is attained from the mixing weight. In several studies, including (Gu et al., 2018; Bachmann et al., 2020; Lou et al., 2020), researchers use (products of) constant curvature Riemannian spaces for graph node embedding where the spaces are assumed to be spherical, hyperbolic, or Euclidean. The work (Xiong et al., 2022) considers a special pseudo-Riemannian manifold named pseudo-hyperboloid that is of constant nonzero curvature and is diffeomorphic to the product manifolds of a unit sphere and the Euclidean space.

Embedding nodes in the aforementioned *restricted* (pseudo-)Riemannian manifolds is achieved through the exponential map in closed forms, which is essentially a geodesic curve on the manifolds as the projected curve of the *cogeodesic orbits* on the manifolds' cotangent bundles (Lee, 2013; Klingenberg, 2011). In our work, we propose to embed the nodes, via more general *Hamiltonian orbits*, into a general manifold, which generalizes the above graph node embedding works.

Manifolds have a diverse set of applications in physics, and their usage and development can be found interwoven throughout the literature. From the physics perspective, the cotangent bundles are the natural phase spaces in classical mechanics (De León & Rodrigues, 2011) where the physical system evolves according to the basic laws of physics modeled as differential equations on the phase spaces. In

[*]Equal contribution [1]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore [2]C3 AI, Singapore. Correspondence to: Qiyu Kang <kang0080@e.ntu.edu.sg>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]If the δ-hyperbolicity distribution is more concentrated at lower values, the more hyperbolic the graph dataset.

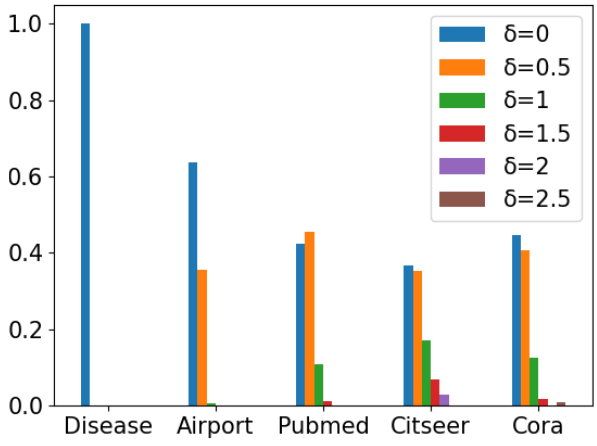


Figure 1. Gromov δ-hyperbolicity distribution of datasets.

this paper, we propose a new GNN paradigm based on Hamiltonian mechanics (Goldstein et al., 2001) with flexible Hamiltonian functions. Our objective is to design a new node embedding strategy that can automatically learn, without extensive tuning, the underlying geometry of any given graph dataset even if it has diverse geometries. We enable the node features to evolve on the manifold under the influence of neighbors. The learnable Hamiltonian function on the manifold guides the node embedding evolution to follow a learnable law analogous to basic physical laws.

Main contributions. Our main contributions are summarized as follows:

1. We consider the graph node embedding problem on an underlying manifold and enable node embedding through a learnable Hamiltonian orbit associated with the Hamiltonian scalar function on the manifold's cotangent bundle.
2. Our node embedding strategy can automatically learn, without extensive tuning, the underlying geometry of any given graph dataset even if it has diverse geometries. We empirically demonstrate its ability by testing on two graph node embedding downstream tasks: node classification and link prediction.
3. From empirical experiments, we observe that the over-smoothing problem of GNNs can be mitigated if the node features evolve through Hamiltonian orbits. By the conservative nature of the Hamiltonian equations, our model enables a stable training and inference process while updating the node features over time and layers.

2. Related Work

While our paper is related to Hamiltonian neural networks in the literature, we are the first, to our best knowledge, to model graph node embedding with Hamiltonian equations. We briefly review Hamiltonian neural networks, Riemannian manifold GNNs, and physics-inspired GNNs in Appendix A.

Notations: We use the *Einstein summation convention* (Lee, 2013) for expressions with tensor indices. When using this convention, if an index variable appears twice in a term, once as a superscript and once as a subscript, it means a summation of the term over all possible values of the index variable. For example, $a^i b_i \triangleq \sum_{i=1}^{d} a^i b_i$.

3. Motivations and Preliminaries

In this section, we briefly review the concepts of the geodesic curve on a (pseudo-)Riemannian manifold from the principle of stationary action in the form of Lagrange's equations. We then further generalize the geodesic curve to the Hamiltonian orbit associated with an energy function H, which is a conserved quantity along the orbit. Our primary goal is to develop a more flexible and robust method for graph node embedding by leveraging the concepts of geodesic curves and Hamiltonian orbits on manifolds. We first summarize the motivation of our work as follows.

Motivation I: from the exponential map to the Riemannian geodesic. The geodesic curve gives rise to the exponential map that maps points from the tangent space to the manifold and has been utilized in (Chami et al., 2019; Bachmann et al., 2020; Xiong et al., 2022) to enable graph node embedding in some restricted (pseudo-)Riemannian manifolds where closed forms of the exponential map are obtainable. From this perspective, by using the geodesic curve, we generalize the graph node embedding to an arbitrary pseudo-Riemannian manifold with learnable local geometry g using Lagrange's equations.

Motivation II: from geodesic to Hamiltonian orbit. Despite the above conceptual generalization for node embedding using geodesic curves, the specific curve formulation involving minimization of curve length may result in a loss of generality for node feature evolution along the curve. We thus further generalize the geodesic curve to the Hamiltonian orbit associated with a more general energy function H that is conserved along the orbit. In Section 4, we propose graph node embedding without an explicit metric by using Hamiltonian orbits with learnable energy functions H.

We kindly refer readers to Appendix B for a more comprehensive elucidation that may enhance their understanding after we introduce the related concepts in Section 3.2.

3.1. Manifold and Riemannian Metric

Manifold and local chart representation. On a d-dimensional manifold M, for each point on M, there exists a triple $\{q, U, V\}$, called a chart, such that U is an open

neighborhood of the point in M, V is an open subset of $\mathbb{R}^d$, and $q : U \to V$ is a homeomorphism, which gives us a coordinate representation for a local area in M.

Tangent and cotangent vector spaces. For any point q on M (we identify each point covered by a local chart on M by its representation q), we may assign two vector spaces named the tangent vector space T_qM and cotangent vector space T_q^*M. The vectors from the tangent and cotangent spaces can be interpreted as representing the velocity and generalized momentum, respectively, of an object's movement in classical mechanics.

Riemannian metric. A Riemannian manifold is a manifold M equipped with a *Riemannian metric* g, where we assign to any point $q \in M$ and pair of vectors $u, v \in T_qM$ an *inner product* $\langle u, v\rangle_{g(q)}$. This assignment is assumed to be smooth with respect to the base point $q \in M$. The length of a tangent vector $u \in T_qM$ is then defined as

$$\|u\|_{g(q)} := \langle u, u\rangle_{g(q)}^{1/2}. \tag{1}$$

- **Local coordinates representation:** In local coordinates with $q = \left(q^1, \ldots, q^d\right)^\intercal \in M, u = \left(u^1, \ldots, u^d\right)^\intercal \in T_qM$ and $v = \left(v^1, \ldots, v^d\right)^\intercal \in T_qM$, the Riemannian metric $g = g(q)$ is a real symmetric *positive definite* matrix and the inner product above is given by

$$\langle u, v\rangle_{g(q)} := g_{ij}(q)u^iv^j \tag{2}$$

- **Pseudo-Riemannian metric:** We may generalize the Riemannian metric to a metric tensor that only requires a *non-degenerate* condition (Lee, 2018) instead of the stringent positive definiteness condition in the inner product. One example of a pseudo-Riemannian manifold is the Lorentzian manifold, which is important in applications of general relativity.

3.2. Geodesic Curves and Exponential Maps

Length and energy of a curve. Let $q : [a, b] \to M$ be a smooth curve.[2] We use $\dot{q}$ and $\ddot{q}$ to denote the first and second order derivatives of $q(t)$, respectively. We define the following:

• length of the curve:

$$\ell(q) := \int_a^b \|\dot{q}(t)\|_{g(q(t))}\,\mathrm{d}t. \tag{3}$$

• energy of the curve:

$$E(q) := \frac{1}{2}\int_a^b \|\dot{q}(t)\|^2_{g(q(t))}\,\mathrm{d}t. \tag{4}$$

[2]We abuse notations in denoting the chart coordinate map as q and the curve as $q(t)$. It will be clear from the context which one is being referred to.

Geodesic curves. On a Riemannian manifold, geodesic curves are defined as curves that have a minimal length as given by (3) and with two fixed endpoints $q(a)$ and $q(b)$. However, computations based on minimizing the length to obtain the curves are difficult. It turns out that the minimizers of $E(q)$ also minimize $\ell(q)$ (Malham, 2016). Consequently, the geodesic curve formulation may be obtained by minimizing the energy of a smooth curve on M.

Principle of stationary action and Euler–Lagrange equation. The Lagrangian function $L(q(t), \dot{q}(t))$ minimizes the following functional (in physics, the functional is known as an **action**)

$$S(q) = \int_a^b L(q(t), \dot{q}(t))\mathrm{d}t \tag{5}$$

with two fixed endpoints at $t = a$ and $t = b$ only if the following *Euler–Lagrange equation* is satisfied:

$$\frac{\partial L}{\partial q^i}(q(t), \dot{q}(t)) - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}^i}(q(t), \dot{q}(t)) = 0. \tag{6}$$

Geodesic equation for geodesic curves. The Euler–Lagrange equation derived from minimizing the energy (4) with local coordinates representation,

$$L = \frac{1}{2}\|\dot{q}(t)\|^2_{g(q(t))} = \frac{1}{2}g_{ik}(q)\dot{q}^i\dot{q}^k \tag{7}$$

is expressed as the following ordinary differential equations called the *geodesic equation*:

$$\ddot{q}^i + \Gamma^i_{jk}\dot{q}^j\dot{q}^k = 0, \tag{8}$$

for all $i = 1, \ldots, d$, where the Christoffel symbols $\Gamma^i_{jk} := \frac{1}{2}g^{i\ell}\left(\frac{\partial g_{\ell j}}{\partial q_k} + \frac{\partial g_{k\ell}}{\partial q_j} - \frac{\partial g_{jk}}{\partial q_\ell}\right)$ and $[g^{ij}]$ denotes the inverse matrix of the matrix $[g_{ij}]$. The solutions to the geodesic equation (8) give us the geodesic curves.

Exponential map. Given the geodesic curves, at each point $q \in M$, for velocity vector $v \in T_qM$, the *exponential map* is defined to obtain the point on M reached by the unique geodesic that passes through q with velocity v at time $t = 1$ (Lee, 2018). Formally, we have

$$\exp_q(v) = \gamma(1) \tag{9}$$

where $\gamma(t)$ is the curve given by the geodesic equation (8) with initial conditions $q(0) = q$ and $\dot{q}(0) = v$.

With regards to **Motivation I**, we note that (Chami et al., 2019) considers graph node embedding over a homogeneous negative-curvature Riemannian manifold called hyperboloid manifold. In contrast, we generalize the embedding of nodes to an arbitrary pseudo-Riemannian manifold through the geodesic equation (8) with a learnable metric g that derives the local graph geometry from the nodes and their neighbors.

3.3. From Geodesics to General Hamiltonian Orbits

The geodesic curves and the derived exponential map essentially come from (5) with L in (7) specified from the curve energy (4). However, the curves derived from this specific action may potentially sacrifice efficacy for the graph node embedding task since we do not know what a reasonable action formulation that guides the evolution of the node feature in this task is. Therefore, we follow the principle of stationary action but consider a learnable action that is more flexible than the length or energy of the curve. To better model the conserved quantity during the feature evolution, we reformulate the Lagrange equation to the Hamilton equation. This is our **Motivation II**.

Hamiltonian function and equations. The Hamiltonian orbit $(q(t), p(t))$ is given by the following *Hamiltonian equations* with a *Hamiltonian function* H:

$$\dot{q}^i = \frac{\partial H}{\partial p_i}, \quad \dot{p}_i = -\frac{\partial H}{\partial q^i}, \tag{10}$$

where q is the local chart coordinate on the manifold while p can be interpreted as a vector of generalized momenta in the cotangent vector space. In classical mechanics, the $2d$-dimensional pair (q, p) is called *phase space* coordinates that fully specify the state of a dynamic system with p guiding the movement direction and speed. Later, we consider the node feature evolution following the trajectory specified by the phase space coordinates.

Hamiltonian function vs. Lagrangian function. The Hamiltonian function can be taken as the Legendre transform of the Lagrangian function:

$$\begin{aligned} H(q,p) &= p_i\dot{q}^i - L(q,\dot{q}) \\ \text{with } \dot{q} = \dot{q}(p) \text{ s.t. } p &= \frac{\partial L}{\partial \dot{q}}. \end{aligned} \tag{11}$$

If H is restricted to strictly convex functions, the Hamiltonian formalism is equivalent to a Lagrangian formalism (De León & Rodrigues, 2011).

Geodesic equation reformulated as Hamiltonian function. If H is set as

$$H(q,p) = \frac{1}{2} g^{ij}(q) p_i p_j, \tag{12}$$

where $[g^{ij}]$ denotes the inverse matrix of the matrix $[g_{ij}]$, we have the following Hamiltonian equations:

$$\dot{q}^i = g^{ij} p_j, \quad \dot{p}_i = -\frac{1}{2}\partial_i g^{jk} p_j p_k. \tag{13}$$

The Hamiltonian orbit $(q(t), p(t))$, as the solution of (13), gives us again the *geodesic curves* $q(t)$ on the manifold M if we only look at the first d-dimensional coordinates.

Theorem 1 (Conservation of energy (Da Silva & Da Salva, 2008))**.** *$H(q(t), p(t))$ is constant along the Hamiltonian orbit as solutions of* (10).

In physics, H typically represents the total energy of a system, and Theorem 1 indicates that the time evolution of the system follows the law of conservation of energy.

4. Proposed Framework

We consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of a finite set $\mathcal{V}$ of vertices, together with a subset $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ of edges. Our objective is to learn a mapping f that maps node features to embedding vectors:

$$f(\mathcal{V}, \mathcal{E}) = Z \in \mathbb{R}^{|\mathcal{V}| \times d}.$$

The embedding Z is supposed to capture both semantic and topological information. Since the input node features in most datasets are sparse, a fully connected (FC) layer is first applied to compress the raw input node features. Let ${}^n q$ be the d-dimensional compressed node feature for node n after the FC layer.[3] However, empirical experiments (see "MLP" results in Section 5.1) indicate that for the graph node embedding task, such simple raw compressing without any consideration of the graph topology does not produce a good embedding. Further graph neural network layers are thus required to update the node embedding.

We consider the node features $\{{}^n q\}_{n \in \mathcal{V}}$ to be located on an embedding manifold M and take the node features as the chart coordinate representations for points on the manifold. In **Motivations I and II** in Section 3, we have provided the rationale for generalizing graph node embedding from the exponential map to the pseudo-Riemannian geodesic, and further to the Hamiltonian orbit. To enforce the graph node feature update on the manifold with well-adapted learnable local geometry, we make use of the concepts from Section 3.

4.1. Model Architecture

Node feature evolution along Hamiltonian orbits in a Hamiltonian layer. As introduced in Section 3.3, the $2d$-dimensional *phase space* coordinates (q, p) fully specify a system's state. Consequently, for the node feature as a point on the manifold M, we associate to each point ${}^n q$ a *learnable momentum vector* ${}^n p$. This vector guides the direction and speed of the feature update, allowing the node feature to evolve along Hamiltonian orbits on the manifold. More specifically, we set

$${}^n p = Q_{\text{net}}({}^n q) \tag{14}$$

where Q_{net}[4] is instantiated by an FC layer. We consider a learnable Hamiltonian function H_{net} that specifies the node feature evolution trajectory in the phase space via the

[3] We put the node index to the left of the variable to distinguish it from the manifold dimension index.

[4] The subscript "net" of a function F_{net} indicates that the function F is parameterized by a neural network.

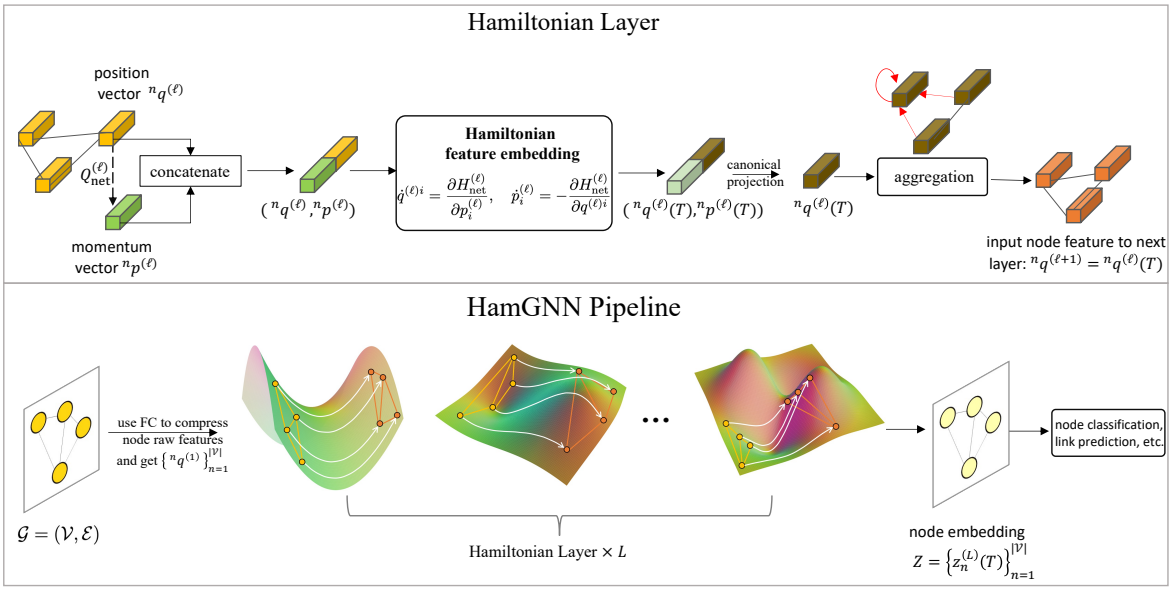


Figure 2. HamGNN architecture: in each layer, each node is assigned a learnable "momentum" vector ${}^np$ (cf. (14)) at time $t = 0$, which initializes the evolution of the node feature. The node features evolve on a manifold following (15) to $({}^nq(T), {}^np(T))$ at the time $t = T$. We only take ${}^nq(T)$ as the embedding and input it to the next layer. After L layers, we take ${}^nq^{(L)}(T)$ as the final node embedding.

Hamiltonian equations:

$$\dot{q}^i = \frac{\partial H_{\text{net}}}{\partial p_i}, \quad \dot{p}_i = -\frac{\partial H_{\text{net}}}{\partial q^i} \tag{15}$$

with learnable Hamiltonian energy function

$$H_{\text{net}} : (q, p) \mapsto \mathbb{R}. \tag{16}$$

The node features are updated along the Hamiltonian orbits, which are curves starting from each node $({}^nq, {}^np)$ at $t = 0$. In other words, they are the solution of (15) with the initial conditions $({}^nq(0), {}^np(0)) = ({}^nq, {}^np)$ at $t = 0$. The solution of (15) on the phase space for each node $n \in \mathcal{V}$ at time T is given by a differential equation solver (Chen et al., 2018a), and denoted by $({}^nq(T), {}^np(T))$. The canonical projection $\pi({}^nq(T), {}^np(T)) = {}^nq(T)$ is taken to obtain the node feature embeddings on the manifold at time T. The aforementioned operations are performed within one layer, and we call it the *Hamiltonian layer*.

Neighborhood Aggregation. After the node features update along the Hamiltonian orbits, we perform neighborhood aggregation on the features $\{{}^nq^{(\ell)}(T)\}_{n\in\mathcal{V}}$, where ℓ indicates the ℓ-th layer. Let $\mathcal{N}(n) = \{m : (n, m) \in \mathcal{E}\}$ denote the set of neighbors of node $n \in \mathcal{V}$. We only perform a simple yet efficient aggregation (see Section 5) for node n as follows:

$${}^nq^{(\ell+1)} = {}^nq^{(\ell)}(T) + \frac{1}{|\mathcal{N}(n)|} \sum_{m\in\mathcal{N}(n)} {}^mq^{(\ell)}(T). \tag{17}$$

Layer stacking for local geometry learning. We stack up multiple Hamiltonian layers with neighborhood aggregation in between them. We first give an intuitive explanation for the case where H_{net} is set as (12). A learnable metric g_{net} for the manifold is involved (see Section 4.2.1 for more details) and the features are evolved following the geodesic curves with minimal length (see Section 3). Within each Hamiltonian layer, the metric g_{net} that is instantiated by a smooth FC layer only depends on the local node position on a pseudo-Riemannian manifold that varies from point to point. Note that with layer stacking, these features contain information aggregated from their neighbors. The metric g_{net}, therefore, learns from the graph topology, and each node is embedded with a local geometry that depends on its neighbors. In contrast, (Chami et al., 2019; Bachmann et al., 2020; Xiong et al., 2022) consider graph node embedding using geodesic curves over some fixed manifold without adjustment of the local geometry.

At the beginning of Section 4, we have assumed the node features $\{{}^nq\}_n$ to be located in local charts of a preliminary embedding manifold M. The basic philosophy is that the embedding manifold evolves with a metric structure that adapts successively with neighborhood aggregation along multiple layers, whereas each node's features evolve to the most appropriate embedding on the manifold along the curves. For a general learnable H_{net}, the Hamiltonian orbit that starts from one node has aggregated information from its neighbors, which guides the learning of the curve that the node will be evolved along. Therefore, each node is embedded into a manifold with adaptation to the underlying geometry of any given graph dataset even if it has diverse geometries.

Conservation of H_{net}. From Theorem 1, the feature updating through the orbit indicates that the H_{net} is conserved along the curve.

Model summary. Our model is called *HamGNN* as we use Hamiltonian orbits for node feature updating on the manifold. We summarize the HamGNN model architecture in Figure 2 and Algorithm 1. The forms of the Hamiltonian function H_{net} are given in Section 4.2.

4.2. Different Hamiltonian Orbits

We next propose different forms for H_{net} from which the corresponding Hamiltonian orbit and its variations are obtained in our GNN model. The node features are updated along the Hamiltonian orbits, which are curves starting from each node $({}^n q, {}^n p)$ at $t = 0$. Beginning with Motivation I from the paper, we design a learnable metric g_{net} in (12) in Section 4.2.1. This approach relaxes the curve formulation constraint used in (Chami et al., 2019; Bachmann et al., 2020; Xiong et al., 2022) and enables learnable geodesic curves to guide feature evolution on arbitrary (pseudo-)Riemannian manifolds while learning local geometry from graph datasets. To further extend learnable geodesic curves to learnable Hamiltonian orbits on manifolds, as stated in Motivation II, we introduce a flexible H instantiated by an FC layer without constraints in Section Section 4.2.2. Subsequently, we present variations based on Section 4.2.2 in Sections 4.2.3 through 4.2.5 to examine their performance. In Section 4.2.3, we add constraints to ensure that H is a convex function, which allows us to equivalently test a more restricted Lagrangian formalism. In Section 4.2.4, we include less restricted Hamiltonian mechanics without strict constant energy constraints. Finally, in Section 4.2.5, we consider a more flexible representation using the symplectic 2-form in comparison to Section 4.2.2.

4.2.1. Learnable Metric g_{net}

In this subsection, we consider node embedding onto a pseudo-Riemannian and set H_{net} as (12) where a learnable metric g_{net} for the manifold is involved. Within each Hamiltonian layer, the metric g_{net} instantiated by a smooth FC layer depends on the local node position on the pseudo-Riemannian manifold that varies from point to point. The output of g_{net} at position q represents the inverse metric local representation $[g^{ij}]$. However, from (12), the space complexity is order d^3 due to the partial derivative of g's output being a $d \times d$ matrix. We therefore only consider *diagonal metrics* to mitigate the space complexity. More specifically, we now define

$$g_{\text{net}}(q) = \text{diag}([\underbrace{-1, \ldots, -1}_{r}, \underbrace{1, \ldots, 1}_{s}] \odot h_{\text{net}}(q)) \quad (18)$$

where $h_{\text{net}} : \mathbb{R}^d \to \mathbb{R}^d$ consists of non-linear trainable layers and $\odot$ denotes element-wise multiplication. To ensure non-degeneracy of the metric, the output of h_{net} is set to be away from 0 with the final activation function of it being strictly positive. The vector $[-1, \ldots, -1, 1, \ldots, 1]$ controls the signature (r, s) (Lee, 2018) of the metric g with $r + s = d$, where r and s are the number of -1s and 1s, respectively. The signature of the metric is set to be a hyperparameter. According to (13), we have

$$\dot{q}^i = g_{\text{net}}^{ij} p_j, \quad \dot{p}_i = -\frac{1}{2} \partial_i g_{\text{net}}^{jk} p_j p_k. \quad (19)$$

Intuitively, the node features evolve through the "shortest" curves on the manifold. The exponential map used in (Chami et al., 2019; Bachmann et al., 2020; Xiong et al., 2022) is essentially the geodesic curve on a hyperbolic manifold with an explicit formulation due to the manifold type restriction. We do not enforce any assumption here and let the model learn the embedding geometry.

4.2.2. Learnable H_{net}

Different from Section 4.2.1 where H is set as (12) with a pseudo-Riemannian metric, we choose a more flexible H instantiated by an FC layer and consider the Hamiltonian equations:

$$\dot{q}^i = \frac{\partial H_{\text{FC}}}{\partial p_i}, \quad \dot{p}_i = -\frac{\partial H_{\text{FC}}}{\partial q^i}. \quad (20)$$

4.2.3. Learnable Convex H_{net}

As discussed in Section 3.3, if H_{net} is restricted to strictly convex functions, the Hamiltonian formalism can degenerate to a *Lagrangian formalism* through the Legendre transformation (11). We take the following restricted Hamiltonian equations

$$\dot{q}^i = \frac{\partial H_{\text{net}}}{\partial p_i}, \dot{p}_i = -\frac{\partial H_{\text{net}}}{\partial q^i}, \text{s. t. } H_{\text{net}} \text{ is convex}, \quad (21)$$

where a stationary action in (5) is achieved. To guarantee that H_{net} is convex, we follow the work in (Amos et al., 2017) to set non-negative layer weights from the second layer in H_{net}, and all activation functions in H_{net} to be convex and non-decreasing. This network design is shown to be able to approximate any convex functions in (Chen et al., 2018b).

4.2.4. Learnable H_{net} with Relaxation

Different from Section 4.2.2, we enforce additional system biases along the curve as follows:

$$\dot{q}^i = \frac{\partial H_{\text{net}}}{\partial p_i}, \quad \dot{p}_i = -\frac{\partial H_{\text{net}}}{\partial q^i} + f_{\text{net}}(q). \quad (22)$$

Instead of keeping the energy during the feature update along the Hamiltonian orbit, we now also include an additional energy term during the node feature update.

Algorithm 1 Graph Node Embedding with HamGNN

Initialize: the network modules including Hamiltonian function network $\{H_{\text{net}}^{(\ell)}\}_{\ell=1}^{L}$, the learnable momentum function $\{Q_{\text{net}}^{(\ell)}\}_{\ell=1}^{L}$, and the raw node features compressor network FC.
I. Training:
for Epoch 1 **to** N **do**
  1). At each epoch, perform the following to obtain the embedding Z with the n-th column being ${}^{n}q^{(L)}(T)$:
  Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with raw node features.
  Apply FC to compress raw node features and get $\{{}^{n}q^{(1)}\}_{n=1}^{|\mathcal{V}|}$.
  for layer $\ell = 1$ **to** L **do**
    i). Solve the following Hamiltonian equations (15) (or its variants) using neural differential equation solvers (Chen et al., 2018a):

$$\dot{q}^{(\ell)i} = \frac{\partial H_{\text{net}}^{(\ell)}}{\partial p_i^{(\ell)}}, \quad \dot{p}_i^{(\ell)} = -\frac{\partial H_{\text{net}}^{(\ell)}}{\partial q^{(\ell)i}}$$

    with $q^{(\ell)}(0) = {}^{n}q^{(\ell)}$ and $p^{(\ell)}(0) = Q_{\text{net}}^{(\ell)}({}^{n}p^{(\ell)})$ at $t = 0$. The feature vector ${}^{n}q^{(\ell)}(T)$ at time T is obtained from this step.
    ii). Perform neighborhood aggregation according to the following equation (17) to obtain ${}^{n}q^{(\ell+1)}$:

$${}^{n}q^{(\ell+1)} = {}^{n}q^{(\ell)}(T) + \frac{1}{|\mathcal{N}(n)|} \sum_{m \in \mathcal{N}(n)} {}^{m}q^{(\ell)}(T)$$

    which is used as the initial condition at the next layer $\ell + 1$.
  end for
  2). Utilize backpropagation to minimize the cross-entropy loss in node classification and link prediction, with the latter employing negative sampling.
  3). Perform validation over the validation split.
  4). Save the model parameters.
end for
II. Testing:
Load the model from the best validation epoch and perform **Step I.1).** to obtain the final embedding over the test split. Perform node classification or link prediction.

4.2.5. Learnable H_{net} with a flexible symplectic form

Hamiltonian equations have a more flexible representation using the symplectic 2-form (cf. Appendix G). The chart coordinate representation (q, p) may not be the Darboux coordinate system for the symplectic 2-form. Even if our learnable H_{net} may be able to learn the energy representation under the chosen chart coordinate system, we consider a learnable symplectic 2-form to act in concert with H_{net}. More specifically, following (Chen et al., 2021), we have the following 1-form

$$\theta_{\text{net}}^{1} = f_{i,\text{net}} \mathrm{d}q^i,$$

where $f_{i,\text{net}} : M \to \mathbb{R}^d$ is the output's i-th component of the neural network parameterized function. The Hamiltonian orbit is then provided by (Chen et al., 2021) as follows:

$$\left(\dot{q}^i, \dot{p}^i\right) = W^{-1}(q,p) \nabla H_{\text{net}}(q,p), \tag{23}$$

where the skew-symmetric $2d \times 2d$ matrix W, whose elements are written in terms of $(\partial_i f_{j,\text{net}} - \partial_j f_{i,\text{net}})$, is given in (24) in the appendix due to space limitations.

5. Experiments

In this section, we implement the proposed HamGNNs with different settings as shown in Section 4.2 and Appendix C. We select **datasets with various geometries** including the three citation networks: Cora (McCallum et al., 2004), Citeseer (Sen et al., 2008), Pubmed (Namata et al., 2012); and two low hyperbolicity datasets (Chami et al., 2019): Disease and Airport (cf. Table 5). Furthermore, we create **new datasets with more complex geometry** by combining Disease and Cora/Citeseer so that the new datasets have a mixture of both hyperbolic and Euclidean local geometries. Adhering to the experimental settings of (Chami et al., 2019; Gu et al., 2018; Bachmann et al., 2020; Lou et al., 2020; Xiong et al., 2022), we evaluate the effectiveness of the node embedding by performing *two downstream tasks: node classification and link prediction* using the embeddings. Rather than beating all the existing GNNs on these two specific tasks, we want to demonstrate that our node embedding strategy is able to automatically learn, without extensive tuning, the underlying geometry of any given graph dataset even if its underlining geometry is very complex, e.g., a mixture of multiple geometries. Such examples can be found in Table 3. Due to space constraints, we refer the readers to Appendix E for the description of datasets and implementation details.

To fairly compare the performance of the proposed HamGNN, for the node classification tasks, we select several popular GNN models as the baseline. These include *Euclidean GNNs*: GCN (Kipf & Welling, 2017a), GAT (Veličković et al., 2018), SAGE (Hamilton et al., 2017), and SGC (Wu et al., 2019); *Hyperbolic GNNs* (Chami et al., 2019; Liu et al., 2019): HGNN, HGCN, HGAT and LGCN (Zhang et al., 2021b); a *GNN that mixes Euclidean and hyperbolic embeddings*: GIL (Zhu et al., 2020b); *(Pseudo-)Riemannian GNNs*: κ-GCN (Bachmann et al., 2020) and $\mathcal{Q}$-GCN (Xiong et al., 2022); as well as *Graph Neural Diffusions*: GRAND (Chamberlain et al., 2021b) and GraphCON (Rusch et al., 2022). We also include the MLP baseline, which does not utilize the graph topology information. To further demonstrate the advantage of HamGNN, we also

include one vanilla ODE system, whose formulation is given in Appendix F.1. This vanilla ODE system neither includes the learnable "momentum" vector p nor adheres to the Hamiltonian orbits (26). For the link prediction task, we compare HamGNN to the *standard graph node embedding models*, including all the aforementioned baselines in the node classification task except the graph neural diffusion baselines. For the link prediction tasks, we report the best results from different versions of HamGNN: HamGNN (19) on the Disease dataset and (20) on the remaining datasets.

Method δ-hyperbolicity	Disease 0.0	Airport 1.0	Pubmed 3.5	Citeseer 4.5	Cora 11.0
MLP	50.00±0.00	76.96±1.77	71.95±1.38	58.10±1.87	57.15±1.15
HNN (Ganea et al., 2018)	56.40±6.32	80.49±1.54	71.60±0.47	55.13±2.04	58.03±0.55
GCN (Kipf & Welling, 2017a)	81.10±1.33	82.25±0.56	77.83±0.77	71.78±0.34	80.29±2.29
GAT (Veličković et al., 2018)	87.01±2.77	92.99±0.83	77.58±0.81	68.07±1.31	80.33±0.61
SAGE (Hamilton et al., 2017)	81.60±7.68	81.97±0.85	77.63±0.15	65.90±2.32	74.50±0.88
SGC (Wu et al., 2019)	82.78±0.93	81.40±2.21	76.83±1.11	70.88±1.32	81.98±1.71
HGNN (Liu et al., 2019)	80.51±5.70	84.54±0.72	76.65±1.38	69.43±0.99	79.53±0.98
HGCN (Chami et al., 2019)	89.87±1.13	85.35±0.65	76.38±0.81	65.80±2.04	78.70±0.96
HGAT (Zhang et al., 2021a)	88.68±3.36	87.50±0.99	78.00±0.50	69.20±0.96	80.88±0.75
LGCN (Zhang et al., 2021b)	88.47±1.80	88.22±0.18	77.35±1.38	68.08±1.98	80.60±0.92
GIL (Zhu et al., 2020b)	90.78±0.45	91.52±1.74	77.76±0.57	71.10±1.24	82.10±1.12
κ-GCN (Bachmann et al., 2020)	-	87.92±1.33	**79.20±0.65**	**73.25±0.51**	81.08±1.45
$\mathcal{Q}$-GCN (Xiong et al., 2022)	70.79±1.23	89.72±0.52	**81.34±1.54**	**74.13±1.41**	**83.72±0.43**
Vanilla ODE in (27)	71.81±18.85	90.34±0.67	73.30±3.31	56.60±1.26	68.38±1.18
GRAND (Chamberlain et al., 2021b)	74.52±3.37	60.02±1.55	**79.32±0.51**	71.76±0.79	**82.80±0.92**
GraphCON (Rusch et al., 2022)	87.50±4.06	68.61±2.10	78.80±0.97	71.33±0.83	82.49±1.08
HamGNN (19)	**91.26±1.40**	**95.50±0.48**	78.08±0.48	70.12±0.86	**82.16±0.80**
HamGNN (20)	88.74±1.17	95.11±0.40	78.18±0.54	71.48±1.42	81.52±1.27
HamGNN (21)	84.57±5.78	93.28±1.40	78.83±0.46	72.00±0.73	81.84±0.88
HamGNN (22)	87.48±5.90	95.46±0.68	78.30±0.34	**72.38±0.85**	81.56±0.97
HamGNN (23)	88.35±1.51	93.66±0.16	78.60±0.32	71.52±1.41	81.24±0.59
HamGNN (25)	**91.18±0.99**	**95.80±0.19**	77.90±0.49	69.18±1.63	80.90±0.35
HamGNN (26)	**91.50±2.07**	**95.99±0.13**	78.26±0.64	69.10±1.95	80.10±1.56

Table 1. Node classification accuracy(%). The best, second best, and third best results for each criterion are highlighted in **red**, **blue**, and **cyan**, respectively. "-" indicates the open source code and/or the result is unavailable.

Method δ-hyperbolicity	Disease 0.0	Airport 1.0	Pubmed 3.5	Citeseer 4.5	Cora 11.0
MLP	**83.37±5.04**	87.04±0.56	88.69±1.59	89.65±1.00	91.07±0.56
HNN (Ganea et al., 2018)	81.37±8.78	86.06±2.08	94.69±0.25	89.83±0.39	92.83±0.76
GCN (Kipf & Welling, 2017a)	60.38±2.51	90.97±0.65	91.37±0.09	93.20±0.28	92.89±0.77
GAT (Veličković et al., 2018)	62.03±1.58	91.05±0.83	91.03±0.67	93.83±0.65	93.34±0.50
SAGE (Hamilton et al., 2017)	68.02±0.43	91.40±0.88	93.61±0.26	93.37±0.88	92.94±0.40
SGC (Wu et al., 2019)	59.83±4.01	89.72±0.82	92.16±0.13	94.78±0.77	93.15±0.22
HGNN (Liu et al., 2019)	60.20±1.14	92.46±0.20	93.09±0.09	90.35±0.57	92.05±0.33
HGCN (Chami et al., 2019)	78.09±2.79	94.28±0.20	**96.79±0.01**	93.60±0.14	94.10±0.05
HGAT (Zhang et al., 2021a)	76.32±3.41	94.64±0.51	**96.86±0.03**	93.45±0.25	94.96±0.36
GIL (Zhu et al., 2020b)	**99.97±0.08**	**97.92±2.64**	91.22±3.25	**95.99±8.89**	**97.78±2.31**
κ-GCN(Bachmann et al., 2020)	-	**96.35±0.62**	96.60±0.32	95.79±0.24	94.04±0.34
$\mathcal{Q}$-GCN (Xiong et al., 2022)	-	96.30±0.22	**96.86±0.37**	**97.01±0.30**	**95.16±1.25**
HamGNN	**99.73±0.26**	**99.99±0.01**	92.15±0.30	**99.99±0.00**	**98.20±1.73**

Table 2. Link prediction ROC(%). The best, second best, and third best results for each criterion are highlighted in **red**, **blue**, and **cyan**, respectively. "-" indicates the open source code and/or the result is not available.

5.1. Performance Results and Ablation Studies

Node classification. The node classification performance on the benchmark datasets using the baseline models and the proposed HamGNNs with different H_{net}s is shown in Table 1. We observe that HamGNN adapts well to all datasets with various geometries. For the datasets such as Cora, Citeseer and Pubmed, which can be well embedded into Euclidean space, HamGNNs achieve comparable performance to the conventional Euclidean GNNs. For example, HamGNN (19) achieves the third-best performance on the Cora. For tree-like graph data Disease and Airport, HamGNNs outperform the other GNNs, including all the hyperbolic GNNs, which are tailored for hyperbolic datasets, and Riemannian GNNs which use a Cartesian product of spherical, hyperbolic, and Euclidean spaces. From the results in Table 1, we can see that Riemannian GNNs κ-GCN and $\mathcal{Q}$-GCN are biased towards embedding data into Euclidean space since they have the best performance on high hyperbolicity datasets but average performance on low hyperbolicity datasets.

Furthermore, the comparison between HamGNNs with the vanilla ODE GNN in (27) without any Hamiltonian mechanism implies the importance of the Hamiltonian layer.

Link prediction. In Table 2, we report the averaged ROC for the link prediction task. We observe HamGNN adapts well to all datasets and is the best performer on the Airport, Citeseer, and Cora.

Comparison between different H_{net}. We compare HamGNNs with different H_{net}s as elaborated in Section 4.2 on the node classification task. In Section 3.3, we argue that the geodesic curves derived from the action of curve length may potentially sacrifice efficacy for the graph node embedding task since we do not know what a reasonable action formulation that guides the evolution of the node feature in this task is. We therefore also include a more flexible H_{net} in (20) with other variations, e.g., (21) to (23). From Table 1, we observe that the original HamGNN (19) shows good adaptations of node embedding to various datasets. Simply using a FC layer for H in (20) has no obvious improvement. Further imposing convexity on H in (21) also has little positive impact on the performance. Including system bias in (22) achieves the best performance on Citeseer among all HamGNNs. The HamGNNs that achieve the best performance on Disease and Airport are (25) and (26). Overall, all HamGNN variants elaborated in Section 4.2 have well-adapted node embedding performance for various datasets even though some variants may perform slightly better. This observation indicates that good geometry adaptations may come from the HamGNN model architecture with the Hamiltonian orbit evolution. This is further verified by the node classification performance from vanilla ODE in (27), which does not have a well-adapted node embedding performance and is designed without the philosophy of Hamiltonian mechanics. Good experiment results in Section 5.2 using both (19) and (20) also provide further evidence to support our conclusion.

We next compare the HamGNNs using (20) and (23). The

difference between those two settings is that the symplectic form in (20) is set to be the special Poincaré 2-form while in (23), the symplectic form is a learnable one. We however observe that the two HamGNNs achieve similar performance and the more flexible symplectic form does not improve the model performance. This may be because of the fundamental Darboux theorem (Lee, 2013) in symplectic geometry, which states that we can always find a Darboux coordinate system to give any symplectic form the *Poincaré 2-form*. The feature compressing FC may have the network capacity to approximate the Darboux coordinate map, while the flexible learnable H_{net} also has the network capacity to get the energy representation under the chosen chart coordinate system.

datasets	GCN	HGCN	GIL	$\mathcal{Q}$-GCN	HamGNN (19)	HamGNN (20)
Air+Cora	75.16±0.65	78.72±0.42	82.04±1.27	90.25±0.81	**94.82±0.78**	93.08±0.55
Air+Cite	70.97±0.34	74.65±0.40	77.83±1.57	87.63±0.28	**90.03±0.29**	88.94±0.75

Table 3. Node classification on the Mixed Geometry Dataset. First row: mixed Airport+Cora dataset; Second row: mixed Airport+Citeseer dataset.

Dataset	Models	3 layers	5 layers	10 layers	20 layers
Cora	GCN	80.29±2.29	69.87±1.12	26.50±4.68	23.97±5.42
	HGCN	78.70±0.96	38.13±6.20	31.90±0.00	26.23±9.87
	HamGNN (21)	**81.84±0.88**	**81.08±0.16**	**81.40±0.44**	**80.58±0.30**

Table 4. Node classification accuracy(%) when increasing the number of layers on the Cora dataset.

5.2. Mixed Geometry Dataset

5.2.1. Mixed Airport + Cora Dataset

To understand the node embedding capacity of HamGNN, we created a new mixed geometry graph dataset by combining the Airport (3188 nodes) and Cora (2708 nodes) datasets into a single large graph with 5896 nodes (cf. Table 5). The results are presented in the first row of Table 3.

This new dataset is composed of a mixture of hyperbolic and Euclidean geometries, as the Airport dataset is known to have a more hyperbolic geometry, and the Cora dataset is known to have a more Euclidean geometry, as shown in Figure 1. We choose these two datasets also because they have a similar number of nodes. To standardize the node feature dimension across datasets, we pad the node features with additional zeros. We do not create any *new* edges in the new dataset so that the new graph contains two disconnected subgraphs: Airport and Cora. We use a 60%, 20%, 20% random split for training, validation, and test sets on this new dataset.

We test our HamGNN against several baselines including GCN, HGCN, GIL, and $\mathcal{Q}$-GCN. The results are shown in Table 3 from which we can observe that, in this more complex geometry setting, HamGNN performs the best.

5.2.2. Mixed Airport + Citeseer Dataset

We also include experiments on another new mixed dataset created using Airport and Citeseer. The combination of these two datasets is the same as the previous one. This new dataset therefore has different geometry in each component from Airport or Citeseer. The results are reported in the second row of Table 3. We observe that our HamGNN still performs the best.

These two experiments show that our model can adapt well to the underlying complex geometry.

5.3. Observation of Resilience to Over-Smoothing

As a side benefit of HamGNN, we observe from Table 4 that if more Hamiltonian layers are stacked, HamGNN is still able to distinguish nodes from difference classes, while the other GNNs suffer severe oversmoothing problem (Chen et al., 2020a). This may be because when updating node features their energy is constrained to be alone in the Hamiltonian orbit. So, if two nodes are distinct in terms of their energy at the input of HamGNN, they will still be distinguishable by their energy at the output of HamGNN, no matter how many times the feature aggregation operation has been implemented.

More Experiments. We kindly refer readers to Appendix F where we include more empirical analysis and visualization.

6. Conclusion

In this paper, we have designed a new node embedding strategy from Hamiltonian orbits that can automatically learn, without extensive tuning, the underlying geometry of any given graph dataset even when multiple different geometries coexist. We have demonstrated empirically that our approach adapts better than popular state-of-the-art graph node embedding GNNs to various graph datasets on two graph node embedding downstream tasks.

Acknowledgments

This research is supported by A*STAR under its RIE2020 Advanced Manufacturing and Engineering (AME) Industry Alignment Fund – Pre Positioning (IAF-PP) (Grant No. A19D6a0053) and the National Research Foundation, Singapore and Infocomm Media Development Authority under its Future Communications Research and Development Programme. The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

References

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Ver Steeg, G., and Galstyan, A. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proc. Int. Conf. Mach. Learn.*, pp. 21–29, 2019.

Amos, B., Xu, L., and Kolter, J. Z. Input convex neural networks. In *Proc. Int. Conf. Mach. Learn.*, pp. 146–155, 2017.

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

Bachmann, G., Bécigneul, G., and Ganea, O. Constant curvature graph convolutional networks. In *Proc. Int. Conf. Mach. Learn.*, 2020.

Bi, W., Du, L., Fu, Q., Wang, Y., Han, S., and Zhang, D. Make heterophily graphs better fit gnn: A graph rewiring approach. *arXiv preprint arXiv:2209.08264*, 2022.

Bishnoi, S., Bhattoo, R., Ranu, S., and Krishnan, N. Enhancing the inductive biases of graph neural ode for modeling dynamical systems. *arXiv preprint arXiv:2209.10740*, 2022.

Bo, D., Wang, X., Shi, C., and Shen, H. Beyond low-frequency information in graph convolutional networks. In *Proc. AAAI Conf. Artif. Intell.*, pp. 3950–3957, 2021.

Chamberlain, B. P., Rowbottom, J., Eynard, D., Di Giovanni, F., Xiaowen, D., and Bronstein, M. M. Beltrami flow and neural diffusion on graphs. In *Advances Neural Inf. Process. Syst.*, 2021a.

Chamberlain, B. P., Rowbottom, J., Goronova, M., Webb, S., Rossi, E., and Bronstein, M. M. Grand: Graph neural diffusion. In *Proc. Int. Conf. Mach. Learn.*, 2021b.

Chami, I., Ying, Z., Ré, C., and Leskovec, J. Hyperbolic graph convolutional neural networks. In *Advances Neural Inf. Process. Syst.*, 2019.

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proc. AAAI Conf. Artif. Intell.*, pp. 3438–3445, 2020a.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *Proc. Int. Conf. Mach. Learn.*, pp. 1725–1735, 2020b.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. In *Advances Neural Inf. Process. Syst.*, 2018a.

Chen, R. T. Q. torchdiffeq, 2018. URL https://github.com/rtqichen/torchdiffeq.

Chen, Y., Shi, Y., and Zhang, B. Optimal control via neural networks: A convex approach. *arXiv preprint arXiv:1805.11835*, 2018b.

Chen, Y., Matsubara, T., and Yaguchi, T. Neural symplectic form: Learning hamiltonian equations on general coordinate systems. In *Advances Neural Inf. Process. Syst.*, 2021.

Chien, E., Peng, J., Li, P., and Milenkovic, O. Adaptive universal generalized pagerank graph neural network. *arXiv preprint arXiv:2006.07988*, 2020.

Da Silva, A. C. and Da Salva, A. C. *Lectures on symplectic geometry*, volume 3575. Springer, 2008.

De León, M. and Rodrigues, P. R. *Generalized Classical Mechanics and Field Theory: a geometrical approach of Lagrangian and Hamiltonian formalisms involving higher order derivatives*. Elsevier, 2011.

Fecko, M. *Differential geometry and Lie groups for physicists*. Cambridge university press, 2006.

Ganea, O., Bécigneul, G., and Hofmann, T. Hyperbolic neural networks. In *Advances Neural Inf. Process. Syst.*, 2018.

Goldstein, H., Poole, C., and Safko, J. *Classical Mechanics*. Addison Wesley, 3 edition, 2001.

Greydanus, S., Dzamba, M., and Yosinski, J. Hamiltonian neural networks. In *Advances Neural Inf. Process. Syst.*, 2019.

Gromov, M. *Hyperbolic Groups*. Springer New York, New York, 1987.

Gu, A., Sala, F., Gunel, B., and Ré, C. Learning mixed-curvature representations in product spaces. In *Proc. Int. Conf. Learn. Representations*, 2018.

Haber, E. and Ruthotto, L. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):1–23, December 2017.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Advances Neural Inf. Process. Syst.*, 2017.

Hartman, P. *Ordinary differential equations*. SIAM, 2002.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs, 2021.

Huang, Y., Yu, Y., Zhang, H., Ma, Y., and Yao, Y. Adversarial robustness of stabilized neural ode might be from obfuscated gradients. In Bruna, J., Hesthaven, J., and Zdeborova, L. (eds.), *Proc. Mathematical and Scientific Machine Learning Conference*, pp. 497–515, 2022.

Ji, F., Lee, S. H., Meng, H., Zhao, K., Yang, J., and Tay, W. P. Leveraging label non-uniformity for node classification in graph neural networks. In *Proc. Int. Conf. Mach. Learn.*, Haiwaii, USA, Jul. 2023.

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

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *Proc. Int. Conf. Learn. Representations*, pp. 1–14, 2017a.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *Proc. Int. Conf. Learn. Representations*, 2017b.

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

Kline, M. *Mathematical Thought from Ancient to Modern Times: Volume 2*, volume 2. Oxford university press, 1990.

Klingenberg, W. P. *Riemannian geometry*, volume 1. Walter de Gruyter, 2011.

Lee, J. M. *Introduction to smooth manifolds*. Springer, London, 2013.

Lee, J. M. *Introduction to Riemannian manifolds*. Springer, London, 2018.

Lee, S. H., Ji, F., and Tay, W. P. SGAT: Simplicial graph attention network. In *Proc. Inter. Joint Conf. Artificial Intell.*, Vienna, Austria, Jul. 2022.

Liu, Q., Nickel, M., and Kiela, D. Hyperbolic graph neural networks. In *Advances Neural Inf. Process. Syst.*, 2019.

Lou, A., Katsman, I., Jiang, Q., Belongie, S., Lim, S.-N., and De Sa, C. Differentiating through the fréchet mean. In *Proc. Int. Conf. Mach. Learn.*, pp. 6393–6403, 2020.

Luan, S., Hua, C., Lu, Q., Zhu, J., Zhao, M., Zhang, S., Chang, X.-W., and Precup, D. Revisiting heterophily for graph neural networks. *arXiv preprint arXiv:2210.07606*, 2022.

Malham, S. J. An introduction to lagrangian and hamiltonian mechanics, 2016.

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

Namata, G. M., London, B., Getoor, L., and Huang, B. Query-driven active surveying for collective classification. In *Workshop Mining Learn. Graphs*, 2012.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

Rudin, W. et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1976.

Rusch, T. K., Chamberlain, B. P., Rowbottom, J., Mishra, S., and Bronstein, M. M. Graph-coupled oscillator networks. In *Proc. Int. Conf. Mach. Learn.*, 2022.

Sanchez-Gonzalez, A., Bapst, V., Cranmer, K., and Battaglia, P. Hamiltonian graph networks with ode integrators. In *Advances Neural Inf. Process. Syst. Workshop on Machine Learning and the Physical Sciences*, 2019.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI Magazine*, 29(3):93, Sep. 2008.

She, R., Kang, Q., Wang, S., Tay, W. P., Guan, Y. L., Navarro, D. N., and Hartmannsgruber, A. Stable architectures for deep neural networks. *IEEE Trans. Image Process.*, May 2023.

Song, Y., Kang, Q., Wang, S., Zhao, K., and Tay, W. P. On the robustness of graph neural diffusion to topology perturbations. In *Advances Neural Inf. Process. Syst.*, 2022.

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

Wang, S., Kang, Q., She, R., Tay, W. P., Hartmannsgruber, A., and Navarro, D. N. RobustLoc: Robust camera pose regression in challenging driving environments. In *Proc. AAAI Conf. Artif. Intell.*, Washington, DC, Feb. 2023a.

Wang, S., Kang, Q., She, R., Wang, W., Zhao, K., Song, Y., and Tay, W. P. HypLiLoc: Towards effective LiDAR pose regression with hyperbolic fusion. In *Proc. Conf. Comput. Vision Pattern Recognition*, Vancouver, Canada, Jun. 2023b.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *Proc. Int. Conf. Mach. Learn.*, 2019.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.*, 32(1):4–24, 2021.

Xiong, B., Zhu, S., Potyka, N., Pan, S., Zhou, C., and Staab, S. Pseudo-riemannian graph convolutional networks. In *Advances Neural Inf. Process. Syst.*, 2022.

Yue, X., Wang, Z., Huang, J., Parthasarathy, S., Moosavinasab, S., Huang, Y., Lin, S. M., Zhang, W., Zhang, P., and Sun, H. Graph embedding on biomedical networks: methods, applications and evaluations. *Bioinformatics*, 36(4):1241–1251, 2019.

Zhang, Y., Wang, X., Shi, C., Jiang, X., and Ye, Y. F. Hyperbolic graph attention network. *IEEE Tran. Big Data*, 63(1), 2021a.

Zhang, Y., Wang, X., Shi, C., Liu, N., and Song, G. Lorentzian graph convolutional networks. In *Proc. Web Conf.*, 2021b.

Zhang, Z., Cui, P., and Zhu, W. Deep learning on graphs: A survey. *IEEE Trans. Knowl. Data Eng.*, 34(1):249–270, Jan 2022.

Zhao, K., Kang, Q., Song, Y., She, R., Wang, S., and Tay, W. P. Graph neural convection-diffusion with heterophily. In *Proc. Inter. Joint Conf. Artificial Intell.*, Macao, China, Aug. 2023.

Zhong, Y. D., Dey, B., and Chakraborty, A. Symplectic ode-net: Learning hamiltonian dynamics with control. In *Proc. Int. Conf. Learn. Representations*, 2020.

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

Zhu, S., Pan, S., Zhou, C., Wu, J., Cao, Y., and Wang, B. Graph geometry interaction learning. In *Advances Neural Inf. Process. Syst.*, 2020b.

A. Related Work

Graph Neural Networks (GNNs) (Yue et al., 2019; Ashoor et al., 2020; Kipf & Welling, 2017b; Zhang et al., 2022; Wu et al., 2021; Lee et al., 2022; She et al., 2023; Wang et al., 2023a; Ji et al., 2023) have exhibited remarkable inferential performance in a broad range of applications. In this section, we provide a succinct review of Hamiltonian Neural Networks, Riemannian Manifold GNNs, and Graph Neural Diffusions.

Hamiltonian neural networks. Among the physics-inspired deep learning approaches, Hamiltonian equations have been applied to conserve an energy-like quantity when training neural networks. The work by (Haber & Ruthotto, 2017) introduces a Hamiltonian-inspired neural ODE to stabilize gradients, avoiding vanishing and exploding issues. Further, (Huang et al., 2022) investigates the adversarial robustness of Hamiltonian ODE. In the physics community, several works propose learning a Hamiltonian function from the observation of systems to simulate physical systems or solve problems in particle physics. Studies such as (Greydanus et al., 2019; Zhong et al., 2020; Chen et al., 2021) train a neural network to infer the Hamiltonian dynamics of a physical system, where the Hamiltonian equations are solved using neural ODE solvers. To better model interactions between physical objects, works like (Bishnoi et al., 2022; Sanchez-Gonzalez et al., 2019) employ a graph network with a Hamiltonian inductive design to capture the dynamics of physical systems from observed trajectories. For instance, to forecast dynamics, (Bishnoi et al., 2022) employs graph networks incorporating Hamiltonian mechanics to learn phase space orbits efficiently. This work demonstrates the effectiveness of Hamiltonian graph neural networks on several dynamics benchmarks, including pendulum systems and spring networks. Compared to traditional neural networks, Hamiltonian (graph) neural networks offer several advantages, such as energy preservation, natural incorporation of conservation laws, and a more interpretable and physically meaningful representation of the system.

In this paper, we are neither simulating a Hamiltonian physics system nor solving a real-world Hamiltonian physics problem. Instead, we apply Hamiltonian equations to generic graph node embedding tasks. Our approach generalizes the Riemannian manifold GNNs and has not been investigated in the above-mentioned works.

Riemannian manifold GNNs. Most GNNs in the literature (Yue et al., 2019; Ashoor et al., 2020; Kipf & Welling, 2017b; Zhang et al., 2022; Wu et al., 2021) embed graph nodes in Euclidean spaces. In what follows, we simply call them (vanilla or Euclidean) GNNs. They perform well on some datasets like the Cora dataset (McCallum et al., 2004) whose Gromov δ-hyperbolicity distributions (Chami et al., 2019) is high. When dealing with datasets whose

δ-hyperbolicities are low (hence embedding should more appropriately be in a hyperbolic space) such as the Disease (Chami et al., 2019) and Airport (Chami et al., 2019) datasets, those GNNs suffer from improper node embedding. To better handle hierarchical graph data, (Liu et al., 2019; Chami et al., 2019; Zhang et al., 2021b; Zhu et al., 2020b; Wang et al., 2023b) propose to embed nodes into a hyperbolic space, thus yielding hyperbolic GNNs. Moreover, (Zhu et al., 2020b) proposes a mixture of embeddings from Euclidean and hyperbolic spaces. This mixing operation relaxes the strong space assumption of using only one type of space for a dataset. In some recent studies, such as (Gu et al., 2018; Bachmann et al., 2020; Lou et al., 2020), the researchers use (products of) *constant curvature* Riemannian spaces for graph node embedding where the spaces are assumed to be spherical, hyperbolic, or Euclidean. In work by (Xiong et al., 2022), a special type of pseudo-Riemannian manifold called the pseudo-hyperboloid, which has constant non-zero curvature and is diffeomorphic to the product of a unit sphere and Euclidean space, has been considered for the same purpose.

In this paper, we embed nodes into a general learnable manifold via the Hamiltonian orbit on its symplectic cotangent bundle. This allows our model to flexibly adapt to the inherent geometry of the dataset.

Graph neural diffusion. Neural Partial Differential Equations (PDEs) (Chamberlain et al., 2021b;a; Song et al., 2022; Zhao et al., 2023) are extensions of neural Ordinary Differential Equations (ODEs) (Chen et al., 2018a; Kang et al., 2021) and have been applied to graph-structured data, where different diffusion schemes are assumed when performing message passing on graphs. To be more specific, the heat diffusion model is assumed in (Chamberlain et al., 2021b) and the Beltrami diffusion model is assumed in (Chamberlain et al., 2021a; Song et al., 2022). (Rusch et al., 2022) models the nodes in the graph as coupled oscillators, i.e., a second-order ODE.

While the above-mentioned graph neural diffusion schemes and our model all use ODEs, there is a fundamental difference between our model and graph neural flows. The graph PDE models wrap the message passing function, e.g., aggregation functions like the one in GCN, and attention-based aggregation functions like the one in GAT, into an ODE function. In contrast, our model treats the node embedding process and node aggregation process as two independent processes: we use the ODE function only to learn a suitable node embedding space which is then followed by a node aggregation step. In summary, the ODE layer in our model is a node embedding layer taking node features as the input, whereas graph PDE layers can be interpreted as node aggregation layers taking node features as well as the graph adjacency matrix as the input.

B. Motivation

Our primary goal is to develop a more flexible and robust method for graph node embedding by leveraging the concepts of geodesic curves and Hamiltonian orbits on manifolds.

The embedding strategies in our work and the literature (Chami et al., 2019; Bachmann et al., 2020; Xiong et al., 2022) can be unified in equation (5) (and its dual equation (11)):

$$S(q) = \int_a^b L(q(t), \dot{q}(t))\mathrm{d}t.$$

Our work and the previous works all follow the same "principle of stationary action" (action means "cost", see below for more explanation) to minimize the functional (5). The minimization leads to different curves on fixed or arbitrary manifolds. The embedding strategy is to learn to move the node on the manifolds to specific positions by following these curves.

In previous works (Chami et al., 2019; Bachmann et al., 2020; Xiong et al., 2022), L takes the (7):

$$L = \frac{1}{2}\|\dot{q}(t)\|^2_{g(q(t))} = \frac{1}{2}g_{ik}(q)\dot{q}^i\dot{q}^k$$

where g is fixed to some formulation, depending on whether the fixed manifold is assumed to be spherical, hyperbolic, or Euclidean. These methods learn to move the node on a fixed manifold following limited geodesic curves induced from a fixed g. With the Lagrangian L setting as (7), we obtain the shortest geodesic that induces the exponential map in (pseudo-)Riemannian manifolds, which has been widely used in the literature (Chami et al., 2019; Bachmann et al., 2020; Xiong et al., 2022) to map features to specific manifolds. These successes in the literature demonstrate that the shortest geodesic or more general the "principle of stationary action" helps the node embedding quality.

One philosophical reason for the success, as opined by Pierre Louis Maupertuis, is that "nature is thrifty in all its actions" `https://en.wikipedia.org/wiki/Stationary-action_principle` (Kline, 1990). Here, actions mean "effort" or "cost". The classical physics example is that "light travels between two given points along the path of shortest time". The embedding process (the feature evolution following a geodesic curve) on a manifold also obeys a similar least-action philosophy.

Motivation I: Enhancing the adaptability of node embeddings using Riemannian geodesics. Traditional graph node embedding approaches (Chami et al., 2019; Bachmann et al., 2020; Xiong et al., 2022) often assume a fixed geometry g that induces fixed Riemannian geodesics (i.e., exponential

map) on manifolds for node embedding, which may not adequately represent all types of geometries in graph datasets. By integrating learnable geodesic curves with learnable g on arbitrary (pseudo-)Riemannian manifolds, our method can adaptively learn the local geometry for each dataset. This not only enables the use of a broader class of functions but also offers improved representation for diverse graph data with varying underlying geometries.

In summary, for Motivation I, we replace the fixed g in the aforementioned L with a more flexible and learnable function. This learnable g guides more adaptable geodesic curves on arbitrary (pseudo-)Riemannian manifolds, allowing us to learn the local geometry for each dataset more effectively. As a result, the graph node features that move along these learnable geodesic curves to positions on the manifold will lead to improved embedding quality.

Motivation II: Enhancing node feature evolution with Hamiltonian orbits. While the geodesic curve-based node embedding presented in Motivation I, using (7) with learnable g, allows for a more flexible representation of data, it still has limitations w.r.t. the generality of the node feature evolution along the manifold curve. To tackle this issue, we propose to extend the learnable geodesic curve concept to learnable Hamiltonian orbits on manifolds. These orbits, which are also curves, are associated with a more general Lagrangian function L. Our method can therefore effectively capture the underlying complexities and variations in the data, resulting in better performance in various graph learning tasks.

In summary, Motivation II involves further relaxing the aforementioned L (7) to include more general functions, leading to more general Hamiltonian orbits (which are also curves) on manifolds. Note that geodesic curves are a specific type of Hamiltonian orbit (after the projection). As L and H are dual, we set different learnable H in Section 4.2 of the paper. Graph node features continue to move along the learned curves to positions on the manifold, but the generalized L allows for more flexible curve options than those provided in Motivation I.

C. Some Formulations and More Hamiltonian Orbits

In section, we first present the formulations which are not shown in detail in the main paper due to space constraints. More Hamiltonian-related flows are also presented, which however do not strictly follow the Hamiltonian orbits on the cotangent bundle T^*M.

C.1. W in Section 4.2.5

See (24).

C.2. Learnable Metric g_{net} with Relaxation

Similar to Section 4.2.4, we now impose additional system biases along the curve compared to the cogeodesic orbits Section 4.2.1,

$$\dot{q}^i = g_{\text{net}}^{ij} p_j, \quad \dot{p}_i = -\frac{1}{2}\partial_i g_{\text{net}}^{jk} p_j p_k + f_{\text{net}}(q). \tag{25}$$

Therefore, the projection of the curve from (25) now no longer follows the geodesic curve along the base manifold equipped with metric g_{net}.

C.3. Hamiltonian Relaxation Flow with Higher Dimensional "Momentum"

In the paper main context, we present a new type of Hamiltonian-related flow, which does not strictly follow the Hamiltonian equations. Inspired from the work (Haber & Ruthotto, 2017), we now associate to each node $q \in \mathbb{R}^d$ an additional a learnable momentum vector $p \in \mathbb{R}^k$ which however is not strictly a cotangent vector of the manifold if $d \neq k$. We update the node features using the following equations

$$\begin{aligned} \dot{q} &= \phi\left(h_{\text{net}}^1(p) - \rho q\right), \\ \dot{p} &= \phi\left(h_{\text{net}}^2(q) - \rho p\right). \end{aligned} \tag{26}$$

where h_{net}^1 and h_{net}^2 are neural networks with d-dimensional output and k-dimensional output respectively, ϕ is a non-linear activation function and ρ is a scalar hyper-parameter.

D. Dataset Configuration

The statistics of the datasets we use are reported in Table 5. We follow the data pre-processing strategy in (Chami et al., 2019) to normalize the adjacency matrix and features before inputting them into the GNN models.

Dataset	Nodes	Edges	Classes	Node Features
Disease	1044	1043	2	1000
Airport	3188	18631	4	4
Cora	2708	5429	7	1433
Citeseer	3327	4732	6	3703
Pubmed	19717	44338	3	500

Table 5. Dataset statistics, where the dataset hyperbolicity is shown in Figure 1.

E. Main Paper Experiments Setting

We select the citation networks Cora (McCallum et al., 2004), Citeseer (Sen et al., 2008), and Pubmed (Namata et al., 2012), and the low-hyperbolicity (Chami et al., 2019) Disease, Airport as the benchmark datasets. The citation

$$W = \begin{pmatrix} 0 & \partial_1 f_{2,\text{net}} - \partial_2 f_{1,\text{net}} & \partial_1 f_{3,\text{net}} - \partial_3 f_{1,\text{net}} & \cdots \\ \partial_2 f_{1,\text{net}} - \partial_1 f_{2,\text{net}} & 0 & \partial_2 f_{3,\text{net}} - \partial_3 f_{2,\text{net}} & \cdots \\ \partial_3 f_{1,\text{net}} - \partial_1 f_{3,\text{net}} & \partial_3 f_{2,\text{net}} - \partial_2 f_{3,\text{net}} & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{24}$$

datasets are widely used in graph representation learning tasks. We use the same dataset splitting settings in (Kipf & Welling, 2017a). The low-hyperbolicity datasets Disease and Airport are proposed in (Chami et al., 2019), where the Euclidean GNN models cannot learn the node embeddings effectively. We adhere to the data splitting and preprocessing procedures outlined in (Chami et al., 2019) for the Disease and Airport datasets.

We adjust the model parameters in HamGNN based on the results from the validation data. We use the ADAM optimizer (Kingma & Ba, 2014) with the weight decay as 0.001. We set the learning rate as 0.01 for citation networks and 0.001 for Disease and Airport datasets. The results presented in Table 1 are under the 3 layers HamGNN setting. We report the results by running the experiments over 10 times with different initial random seeds.

HamGNN first compresses the dimension of input features to the fixed hidden dimension (e.g. 64) through a fully connected (FC) layer. Then the obtained hidden features are input to the stacked H_{net} ODE layers and aggregation layers. The q in Hamiltonian flow is initialized by the node embeddings after the FC layer. Table 13 shows the implementation details of the layers in HamGNN.

E.1. ODE solver for Hamiltonian equations

We employ the ODE solver (Chen, 2018) in the implementation of HamGNN. For computation efficiency and performance effectiveness, the fixed-step explicit Euler solver (Chen et al., 2018a) is used in HamGNN. We also compare the influence of ODE solvers and report the results in Table 6. One drawback of the ODE solvers provided in (Chen, 2018) is that they are not guaranteed to have the energy-preserving property in solving the Hamiltonian equations. However, this flaw does not significantly deteriorate our model performance regarding the embedding adaptation to datasets with various structures. Our extensive experiments on the node classification and link prediction tasks have demonstrated that the solvers provided in (Chen, 2018) are sufficient for our use. We leave the use of Hamiltonian equation solvers for future work to investigate whether solvers with the energy-preserving property can better help graph node embedding or mitigate the over-smoothing problem.

F. More Ablation Studies and Experiments

F.1. Vanilla ODE

To demonstrate the advantage of HamGNN's design, we also conduct more experiments that replace the Hamiltonian layer in HamGNN with a vanilla ODE as follows:

$$\dot{q}(t) = \tilde{f}_{\text{net}}(q(t)) \tag{27}$$

where the $\tilde{f}_{\text{net}}(q(t))$ is composed of two FC layers and a non-linear activation function. Compared to the Hamiltonian orbits in Section 4.2, the equation (27) does not include the learnable "momentum" vector for each node and does not follow the Hamiltonian orbits on the cotangent bundle.

F.2. Influence of Metric Signature

We vary the parameter s to explore the influence of the signature of the learned metric g. The results are presented in Table 7. We observe that for the Airport dataset, the influence of the signature is moderate where the best (with $r = 4$) and worst performance (with $r = 2$) has a moderate gap of 0.39% in accuracy. For other datasets, metrics with different pre-defined signatures perform similarly to each other. In Table 1 of the paper, we present the best signature. Further theoretical understanding of the influence of the signature is left for future work.

F.3. Node Classification on Heterophilic Datasets

In this section, we include experiments on the node classification task using heterophilic graph datasets. We would like to emphasize that our HamGNN is not specifically designed for heterophilic datasets, where nodes with different attributes or classes are more likely to be linked together in the graph. Our primary focus is on homophily datasets with varying local geometry, and the experiments presented in the paper mainly address this focus. It is important to note that the local uniform message-passing aggregation (17) used in our paper is also more suited to homophily datasets. The additional experiments on heterophilic datasets in the appendix serve to demonstrate that even in such cases, our model can achieve competitive performance when compared to GNNs that are specifically designed for heterophilic datasets, provided that a good node embedding layer is designed in our paper. In the context of heterophilic datasets, common techniques like higher-order neighbor mixing, adaptive message aggregation, and ego-neighbor separation (Zhu et al., 2020a;

ODE solver	Euler	Euler	Implicit Adams	Implicit Adams	Dopri5
Step size	0.1	0.5	0.1	0.5	-
Cora	81.10±1.13	81.52±1.27	81.62±0.58	81.40±0.77	81.62±0.58

Table 6. Node classification accuracy(%) under different ODE solvers in HamGNN (20).

metric signature (r, s)	Disease	Airport	Pubmed	Citeseer	Cora
$(0, 64)$	90.16±1.15	95.15±0.53	77.82±0.21	69.52±0.51	81.62±1.61
$(1, 63)$	89.45±1.53	95.42±0.54	77.38±0.34	69.98±1.17	**82.16±0.80**
$(2, 62)$	89.84±1.38	95.11±0.40	77.50±0.39	69.26±0.87	81.22±1.56
$(4, 60)$	91.18±1.32	**95.50±0.48**	77.40±0.51	69.12±1.71	81.24±1.97
$(8, 56)$	89.21±3.09	95.19±0.31	77.34±0.39	69.68±0.10	80.98±1.41
$(16, 48)$	88.19±4.92	95.19±0.62	**78.08±0.48**	69.52±0.97	80.86±1.26
$(32, 32)$	**91.26±1.40**	95.27±0.52	77.82±0.22	**70.12±0.86**	81.32±1.06

Table 7. The impact of the signature of metric g in on the node classification performance.

Method	Cornell	Wisconsin	Texas
Geom-GCN	60.54±3.67	64.51±3.66	66.76±2.72
H2GCN	82.70±5.28	87.65±4.98	84.86±7.23
GPRGCN	78.11±6.55	82.55±6.23	81.35±5.32
FAGCN	76.76±5.87	79.61±1.58	76.49±2.87
GCNII	77.86±3.79	80.39±3.40	77.57±3.83
MixHop	73.51±6.34	75.88±4.90	77.84±7.73
WRGAT	81.62±3.90	86.98±3.78	83.62±5.50
GraphCON	75.14±4.95	84.90±2.64	80.00±3.66
HamGNN	76.49±5.10	83.92±4.87	81.62±6.22

Table 8. Node classification accuracy(%) on heterophilic datasets under 10 fixed 48%/32%/20% splits taken from (Pei et al., 2020).

Bo et al., 2021; Abu-El-Haija et al., 2019) hold the potential to enhance our model's performance. We intend to investigate these strategies and their impact on our novel node embedding approach in future work.

In this section, we select the heterophilic graph datasets Cornell, Texas and Wisconsin from the CMU WebKB [5] project where randomly generated splits of data are provided by (Pei et al., 2020). The edges in these graphs represent the hyperlinks between webpages nodes. The labels are manually selected into five classes, student, project, course, staff, and faculty. The features on node are the bag-of-words of the web pages.

For the heterophilic graph datasets, we include the baselines GCN, GAT, SAGE, APPNP (Klicpera et al., 2019), GCNII (Chen et al., 2020b), GPRGNN (Chien et al., 2020), and H2GCN (Zhu et al., 2020a) which are the common baselines for heterophilic graph datasets (Bi et al., 2022). Additionally, we also include GraphCON (Rusch et al., 2022) for comparisons. We report the results by running the experiments over 10 times with different initial random seeds for GraphCON and HamGNN, while for the other baselines on the heterophilic graph datasets, we use the results reported in the paper (Luan et al., 2022). We still observe that our method is competitive for *GNNs that are designed specifically for heterophilic datasets.*

[5] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/

F.4. Aggregation

In the main paper, we have chosen to implement a simple yet effective fixed-weight aggregation in accordance with Occam's razor principle rather than incorporating complex and computationally heavy attention-based aggregation operations. However, our model is compatible with more advanced aggregation methods. To demonstrate this, we have conducted additional experiments using attention-based aggregation with (19), and the results are presented in Table 9. We observe that on the Disease dataset, the attention-based aggregation can further improve the node classification accuracy.

Aggregation	Disease	Airport	Pubmed	Citeseer	Cora
attention	92.72±1.65	94.35±1.00	78.03±0.69	70.53±1.84	81.91±1.03
fixed-weight (17)	91.26±1.40	95.50±0.48	78.08±0.48	70.12±0.86	82.16±0.80

Table 9. Performance of attention-based aggregation on the node classification task.

F.5. Performance on Larger Graph Datasets

In this section, to underscore our model's capacity for handling large graph datasets, we conduct a series of experiments on the Ogbn datasets obtained from https://ogb.stanford.edu/docs/nodeprop/, in compliance with the experimental setup detailed in (Hu et al., 2021). The corresponding results are encapsulated in Table 10. For the training of our model on the Ogbn-products dataset, we employ a straightforward neighborhood sampling approach, as introduced in GraphSage (Hamilton et al., 2017). The GCN results are extracted directly from the leaderboard available at https://ogb.stanford.edu/docs/leader_nodeprop/. We observe that for such large datasets, our model still performs efficiently compared to other methods.

Model	Ogbn-Arxiv	Ogbn-Products
HAMGNN	71.70±0.27	79.87±0.04
GCN	71.74±0.29	75.64±0.21

Table 10. Node classification results(%) on Ogbn datasets

F.6. Computational Cost

To provide further information, we present the memory usage along with the training and inference time in Table 11. When considering efficiency, HamGNN exhibits a reduced training and inference duration in comparison to both HGCN and GIL. Furthermore, it boasts a smaller model size relative to GIL, while maintaining a similar size to GCN/HGCN. Overall, the table suggests that HamGNN models require higher computational resources than GCN but are more efficient than HGCN and GIL, and can achieve higher accuracy on low hyperbolicity graph learning tasks.

	HamGNN(20)	GCN	HGCN	GIL
Num of para	113025	92231	92231	189740
Model Size (MB)	0.431	0.352	0.352	0.724
Inference Time (ms)	3.245	1.662	4.355	14.332
Training Time (ms)	9.612	6.064	38.766	60.022

Table 11. Model size and computation time. All models are using one hidden layer with a hidden dimension of 64 on the CORA dataset for a fair comparison.

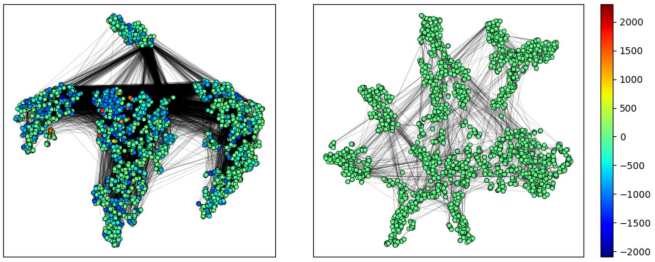
F.7. Examination of learned curvature parameters

In this section, we provide the visualization of the learned Ricci scalar curvature (https://en.wikipedia.org/wiki/Ricci_curvature) that is computed from our learnable metric g. We use one high δ-hyperbolicity dataset, the Cora dataset, and one low δ-hyperbolicity dataset, the Airport dataset. (Note that higher δ-hyperbolicity means *less* hyperbolic.)

Airport: We observe that the local curvatures learned on the Airport dataset show various values, ranging from -2000 to +2000. Few embeddings show positive curvature, and most of the embeddings show **negative or near 0 curvature** at the local geometry. This is expected as hyperbolic GNNs (Chami et al., 2019; Liu et al., 2019) have demonstrated that tree-structured datasets have improved classification accuracy if they are embedded into a hyperbolic space with constant negative curvature rather than the zero curvature Euclidean space. Our model automatically learns to achieve this. The difference is that in HamGNN with (19), the local curvature varies from point to point based on learning, and some points may even be embedded with a local positive curvature geometry. The visualized learned local curvatures explain why our model can surpass the baselines on the lower δ-hyperbolicity datasets.

Cora: For the Cora dataset, from the visualization, we observe that almost all the embeddings are located at locations with **near zero curvature**, i.e., similar to the Euclidean space. This is consistent with our results shown in the paper that HamGNN has a comparable performance with Euclidean-based GNNs on high δ-hyperbolicity datasets. Our HammGNN successfully learns to embed the nodes in spaces that closely resemble Euclidean spaces.

The above curvature visualization further demonstrates that the curvature in HamGNN can successfully adapt for different structures of different datasets, and this learnable metric leads to all-around good performance.



Airport Cora

Figure 3. Ricci scalar curvature on the Cora and Airport dataset using t-SNE.

F.8. Over-Smoothing

Our proposed model aims to alleviate the over-smoothing issue in GNNs during the feature updating stage rather than the aggregation stage. We are not claiming that HamGNN totally resolves the over-smoothing problem. Over-smoothing typically arises from the repeated application of graph convolution or aggregation operations, which tend to mix and average features from neighboring nodes. However, since our model still employs the traditional aggregation stage, our experiments on over-smoothing also suggest that some of the over-smoothing may originate from the feature updating stage. We continue from Section 5.3 to conduct more experiments on the Cora and Pubmed datasets to demonstrate the resilience of HamGNN against over-smoothing.

From Table 12, we observe that if the H_{net} in (21) is convex, HamGNN can even retain its classification ability better than vanilla H_{net} in (20). One possible reason has been indicated in Section 4.2.3 since now the Hamiltonian formalism degenerates to a Lagrangian formalism with a possible minimization of the dual energy functional (5). In physics, lower energy in most cases indicates a more stable system equilibrium. Moreover, to further show that with the convex H_{net} in (21) HamGNN can perform better than the vanilla H_{net} in (20) against over-smoothing, we now include more choices of convex network H_{net} with different layer sizes and different activation functions (as along as the layer weights from the second layer in H_{net} are non-negative, and all activation functions in H_{net} are convex and non-decreasing). The network details of H_{net} are given in Table 13. The experiment results are shown in Table 12. We clearly observe that for different convex functions and on different datasets, HamGNN

with convex H_{net} nearly keeps the full node classification ability even though we have stacked 20 Hamiltonian layers.

Figure 4 shows how the node features evolve over 10 layers. We sample a node from the test set of the PubMed dataset and input it to three models, which are 1) HamGNN, 2) GIL, and 3) GCN. Each of these three GNNs contains 10 layers, and we compute the node feature magnitude, which is defined to be the L_2-norm of the feature vector, and the node feature phase, which is defined to be the cosine similarity between the output feature at the current layer and the input feature to the first layer. We can observe from Figure 4 that HamGNN has its learned node features that change steadily and slowly, while the node features learned by GIL and GCN change abruptly, especially for the feature phases. The features from two nodes of different classes, learned by GIL and GCN, are converging to each other much faster than HamGNN.

In this paper, we have presented extensive experimental analyses supporting HamGNN's ability to mitigate over-smoothing. Further theoretical analysis of our HamGNN is left for future work.

F.9. Stability

A known characteristic of Hamiltonian systems is their energy-conserving nature. Our proposed HamGNN retains this property of stability or conservation. For our analysis, we incorporate three neural ODE-based GNN models: 1) HamGNN, 2) an ODE model employing a positive-definite linear layer as the ODE function, and 3) an ODE model using a negative-definite linear layer as the ODE function. Each of these three GNNs consists of 10 layers. The results, as presented in Figure 5, illustrate noticeable trends: the feature magnitude (analogous to feature energy) learned by HamGNN remains constant across layers, while the feature magnitude learned by the positive-definite model escalates after a few layers and the one learned by the negative-definite ODE model approaches zero. With respect to the feature phase, the phases from two nodes gradually converge when using HamGNN, while those learned by other neural ODEs exhibit abrupt shifts, and the difference between two nodes using the negative-definite ODE is insignificantly small.

The work (Haber & Ruthotto, 2017) constructs a Hamiltonian-inspired neural ODE and asserts its capability to stabilize gradients, thereby circumventing issues of vanishing and exploding gradients. This resilience to vanishing and exploding gradients may also contribute to the resistance to over-smoothing. The further theoretical investigation remains a subject for future work.

G. Differential Geometry and Hamiltonian System

In this supplementary material, we review some concepts from a differential geometry perspective. We hope that this overview makes the paper more accessible for readers from the graph learning community.

G.1. Manifold, Bundles, and Fields

Roughly speaking, a manifold is a topological space that *locally* looks like Euclidean space. More strictly speaking, a topological space $(M, \mathcal{O})$, where $\mathcal{O}$ is the collection of open sets on space M, is called a d-dimensional manifold if for every point $x \in M$, we can find an open neighborhood $U \in \mathcal{O}$ for x and a coordinate map

$$q : U \to q(U) \subseteq \mathbb{R}^d$$

that is a *homeomorphism*, where $\mathbb{R}^d$ is the d-dimensional Euclidean space with the standard topology. (U, q) is called a *chart* of the manifold, which gives us a numerical representation for a local area in M. In this work, we only consider smooth manifolds (Lee, 2013) that any two overlapped charts are smoothly compatible. The set of all smooth functions from M to $\mathbb{R}$ is denoted as $\mathcal{C}^\infty(M)$. On top of the smooth manifolds, we can define other related manifolds like the tangent or cotangent bundles and the more general tensor bundles. From bundles, we can define the vector or covector fields and the more general tensor fields. In this work, we mainly consider the manifold with the 2-forms that are special $(0, 2)$ smooth tensor fields with antisymmetric constraints. More specifically, we mainly consider the *symplectic form* (Lee, 2013) with some other light shed on the metric tensor which is another type of $(0, 2)$ smooth tensor field.

Definition 1 (tangent vector and tangent space)**.** *Let $\gamma : I \to M$ be a smooth curve through $x \in M$ s.t. $\gamma(0) = x$ and I is an interval neighborhood of 0. The* tangent vector *is a directional derivative operator at x along γ that is the* linear map

$$\begin{aligned} v_{\gamma,x} : \mathcal{C}^\infty(M) &\xrightarrow{\sim} \mathbb{R} \\ f &\mapsto (f \circ \gamma)'(0). \end{aligned}$$

We also call the directional derivative operator as "velocity" of γ at 0 and denote it as $\dot{\gamma}(0)$. (More generally, the 'velocity" of γ at t is denoted at $\dot{\gamma}(t)$ which is a tangent vector at point $\gamma(t) \in M$ from the curve reparametrization trick (Fecko, 2006)). Correspondingly, the tangent space *to M at x is the* vector space *over $\mathbb{R}$ with the underlying set*

$$T_x M \coloneqq \{v_{\gamma,x} \mid \gamma \text{ is a smooth curve and } \gamma(0) = x\}. \tag{28}$$

Dataset	Models	3 layers	5 layers	10 layers	20 layers
Cora	GCN	80.29±2.29	69.87±1.12	26.50±4.68	23.97±5.42
	HGCN	78.70±0.96	38.13±6.20	31.90±0.00	26.23±9.87
	HamGNN (20)	81.52±1.27	**81.58±0.73**	79.00±2.17	76.20±0.13
	HamGNN (21)	81.84±0.88	81.08±0.16	**81.40±0.44**	**80.58±0.30**
	HamGNN (21) type 2	**82.10±0.80**	81.10±0.10	81.06±1.49	79.26±1.07
Pubmed	GCN	77.83±0.77	76.00±0.87	77.53±1.06	56.50±12.79
	HGCN	76.38±0.81	77.20±1.05	65.90±9.44	42.16±2.54
	HamGNN (20)	78.18±0.54	77.86±1.45	77.73±1.15	76.13±0.80
	HamGNN (21)	78.83±0.46	78.43±0.25	78.50±0.61	77.20±0.69
	HamGNN (21) type 2	**79.03±0.58**	**78.46±0.11**	**78.53±0.31**	**77.50±0.44**

Table 12. Node classification accuracy(%) when increasing the number of layers on the Pubmed dataset.

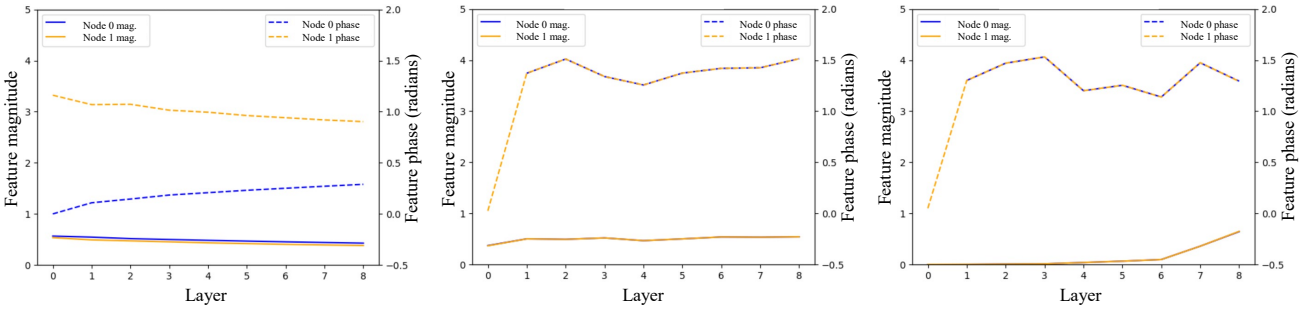


Figure 4. Two nodes from different classes and the evolution of their feature vectors over layers. Left: HamGNN, middle: GIL, right: GCN.

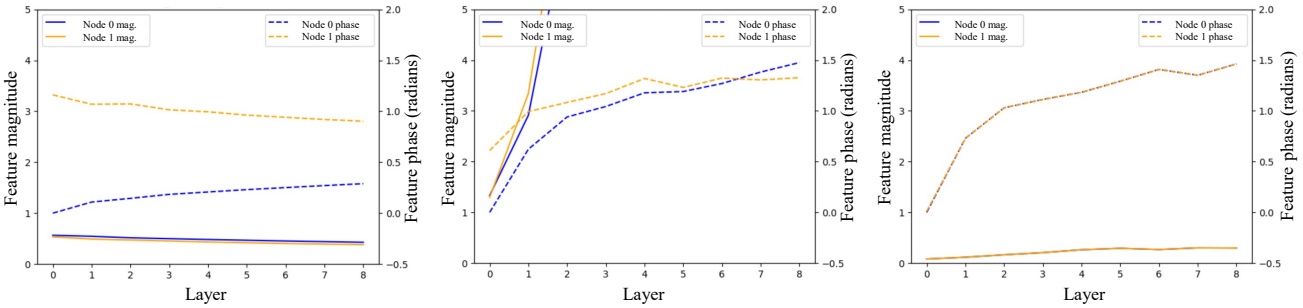


Figure 5. Two nodes from different classes and the evolution of their feature vectors over layers. Left: HamGNN, middle: neural ODE using a positive-definite linear layer, right: neural ODE using a negative-definite linear layer.

Note for $f \in \mathcal{C}^\infty(M)$, using the chart (U, q) with $x \in U$, we have the local representation

$$\begin{aligned} v_{\gamma,x}(f) &:= (f \circ \gamma)'(0) \\ &= (f \circ q^{-1} \circ q \circ \gamma)'(0) \\ &= (q^i \circ \gamma)'(0) \cdot \partial_i \big(f \circ q^{-1}\big)\big|_{q(x)}, \end{aligned}$$

where q^i is the i-th component of q, $\circ$ is the function composition and $(\cdot)|_{q(x)}$ means evaluating $(\cdot)$ at $q(x)$. Therefore for a local chart around x, we have a basis of $T_x M$ as

$$\left\{ \left(\frac{\partial}{\partial q^1}\right)_x, \ldots, \left(\frac{\partial}{\partial q^d}\right)_x \right\}$$

and we call it the chart induced basis, *where* $\left(\frac{\partial}{\partial q^1}\right)_x := \partial_i (f \circ q^{-1})|_{q(x)}$.

Definition 2 (tangent bundle)**.** *Given a smooth manifold M, the* tangent bundle *of M is the* disjoint union *of all the*

Table 13. Neural network modules and the parameters, where $\kappa(x) = 0.5x^2$ if $x > 0$ and $= \exp(x) - 1$ if $x < 0$

Network	Modules	Activation	Output Channels
(20)	Linear tanh Linear	tanh	1
(21)	ReHU(Amos et al., 2017) Linear ReHU Linear ReHU Linear ReHU	ReHU	1
(21) type 2	κ Linear κ Linear κ Linear κ	ReHU	1
(26)	Linear act1 Linear act1	act1	576
(22)	Linear tanh Linear	tanh	1
(25)	Linear tanh Linear Sigmoid	tanh Sigmoid	64
(23)	Linear sin Linear sin	sin	H:1 W:128
Q_{net} Raw feature compressing FC	Linear Linear	identity identity	64 64
MLP in Section 5	Linear ReLU Linear ReLU Linear	ReLU	Number of classes

tangent spaces to M, i.e., we have

$$TM := \coprod_{x \in M} T_x M, \tag{29}$$

equipped with the canonical projection map

$$\tilde{\pi} : TM \to M$$

$$X \mapsto \tilde{\pi}(X),$$

where $\tilde{\pi}(X)$ sends each vector in $T_x M$ to the point x at which it is tangent, and $\coprod$ is the disjoint union. Furthermore, the tangent bundle[6] *is a $2d$-dimensional manifold. If $X \in$*

[6] In this paper, we may just use the word "bundle" to indicate

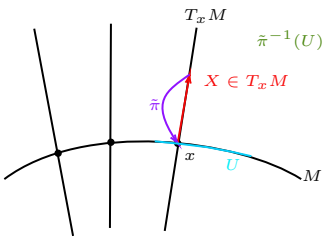


Figure 6. Visualization of a 1-dimensional manifold and its tangent bundle.

$\tilde{\pi}^{-1}(U) \subseteq TM$ with a local chart (U, q) on M s.t. $x \in U$, then $X \in T_{\tilde{\pi}(X)}M$ from the definition. Since $\tilde{\pi}(X) \in U$, X can be written in terms of the chart induced basis*:*

$$X = \tilde{p}^i(X) \left(\frac{\partial}{\partial q^i}\right)_{\tilde{\pi}(X)}, \tag{30}$$

where $\tilde{p}^1, \dots, \tilde{p}^d$ are smooth scalar functions. We can then define the following map as a local chart for the manifold TM induced from the chart (U, q) on M:

$$\begin{aligned}\xi :\ & \tilde{\pi}^{-1}(U) \to q(U) \times \mathbb{R}^d \subseteq \mathbb{R}^{2d} \\ & X \mapsto (q^1(\tilde{\pi}(X)), \dots, q^d(\tilde{\pi}(X)), \tilde{p}^1(X), \dots, \tilde{p}^d(X)),\end{aligned}$$

and the topological structure on TM is derived from the initial topology to ensure continuity.

Definition 3 (vector field)**.** *A vector field on M is a smooth section (Lee, 2013) of the tangent bundle, i.e. a smooth map $\sigma :\ M \to TM$ such that $\tilde{\pi} \circ \sigma = id_M$, where id_M is the identity map on M.*

$$\begin{array}{c} TM \\ \sigma \uparrow \quad \downarrow \tilde{\pi} \\ M \end{array} \tag{31}$$

We denote the set of all vector fields on M by $\Gamma(TM)$:

$$\Gamma(TM) \coloneqq \{\sigma :\ M \to TM \mid \sigma \text{ is smooth and } \tilde{\pi} \circ \sigma = id_M\}. \tag{32}$$

Definition 4 (cotangent vector, cotangent bundle, dual basis, and covector field)**.** *For the vector space T_xM, a continuous linear functional from T_xM to $\mathbb{R}$ is called a cotangent vector at x. The set of all such linear maps is denoted as T_x^*M which is the dual vector space of T_xM. For $f \in \mathcal{C}^\infty(M)$, at each point x, we define the following linear operator in T_x^*M*

$$\begin{aligned}(\mathrm{d}f)_x :\ & T_xM \to \mathbb{R} \\ & X_x \mapsto (\mathrm{d}f)_x(X_x) \coloneqq X_x(f).\end{aligned}$$

Given a chart (U, q) with $x \in U$ and its chart induced basis*, the* dual basis *for the dual space T_x^*M is the set*

$$\left\{(\mathrm{d}q^1)_x, \dots, (\mathrm{d}q^d)_x\right\},$$

where we have $(\mathrm{d}q^a)_x \left(\left(\frac{\partial}{\partial q^b}\right)_x\right) = \left(\frac{\partial}{\partial q^b}\right)_x (q^a) = \delta_b^a$ with $\delta_b^a = 1$ iff $a = b$ and $\delta_b^a = 0$ otherwise. We call it the chart induced dual basis *. Analogous to the above definition of the vector field, we can define the cotangent bundle of M as*

$$T^*M \coloneqq \coprod_{x \in M} T_x^*M \tag{33}$$

which is again a $2d$-dimensional manifold equipped with the canonical projection map

$$\begin{aligned}\pi :\ & T^*M \to M \\ & \omega \mapsto \pi(\omega),\end{aligned} \tag{34}$$

where $\pi(\omega)$ sends each vector in T_x^*M to the point x at which it is cotangent. If $\omega \in \pi^{-1}(U) \subseteq T^*M$ with a local chart (U, q) s.t. $x \in U$, then $\omega \in T_{\pi(\omega)}^*M$ from the definition. Since $\pi(\omega) \in U$, ω can be written in terms of the chart induced dual basis*:*

$$\omega = p_i(\omega)(\mathrm{d}q^i)_{\pi(\omega)}, \tag{35}$$

where $p_1, \dots, p_d$ are smooth scalar functions. We can then define the following map as a local chart for manifold T^*M induced from the chart (U, q) on M:

$$\begin{aligned}\xi :\ & \pi^{-1}(U) \to q(U) \times \mathbb{R}^d \subseteq \mathbb{R}^{2d} \\ & \omega \mapsto (q^1(\pi(\omega)), \dots, q^d(\pi(\omega)), p_1(\omega), \dots, p_d(\omega)),\end{aligned}$$

and the topological structure on T^*M is derived from the initial topology to ensure continuity. The covector fields are smooth sections of T^*M. The set of all covector fields is denoted as $\Gamma(T^*M)$.

Definition 5 (tensor field and 2-form)**.** *The tensor field can be defined using the smooth sections on tensor bundles analogously to the vector fields or the covector fields. We refer readers to (Lee, 2013) for more details. Here, instead of a rigorous definition, we show some basic properties of the tensor fields. For a (r, s) tensor field τ, at $x \in M$, it is a multilinear map*

$$\tau_x :\ \underbrace{T_x^*M \times \cdots \times T_x^*M}_{r \text{ copies}} \times \underbrace{T_xM \times \cdots \times T_xM}_{s \text{ copies}} \to \mathbb{R}.$$

The differential k-form ω is the $(0, k)$ tensor field that admits alternating (Lee, 2013). Specifically, for the 2-form ω, at each point x, ω_x is a antisymmetric $(0, 2)$ *tensor*

$$\omega_x :\ T_xM \times T_xM \to \mathbb{R} \tag{36}$$

$$\text{s. t. } \omega_x(X_1, X_2) = -\omega_x(X_2, X_1)\ \forall\, X_1, X_2 \in T_xM \tag{37}$$

the total space in the bundle.

which in other words, ω satisfies

$$\omega : \Gamma(TM) \times \Gamma(TM) \to \mathcal{C}^\infty(M) \tag{38}$$
$$\text{s.t. } \omega(X, Y) = -\omega(Y, X) \,\forall\, X, Y \in \Gamma(TM) \tag{39}$$

In local chart representation, we have that every k-form ω can be expressed locally *on U as*

$$\omega = \omega_{a_1 \cdots a_k} \mathrm{d}x^{a_1} \wedge \cdots \wedge \mathrm{d}x^{a_k}, \tag{40}$$

where $\omega_{a_1 \cdots a_k} \in \mathcal{C}^\infty(U)$, $1 \le a_1 < \cdots < a_k \le \dim M$ are increasing sequences and $\mathrm{d}x^{a_1} \wedge \cdots \wedge \mathrm{d}x^{a_k}$ is the wedge product (Lee, 2013). Here we could abstractly view the set $\{\mathrm{d}x^{a_1} \wedge \cdots \wedge \mathrm{d}x^{a_k}\}_{a_1,\ldots,a_k}$, with a_i enumerated from 1 to d, abstractly as a basis without more illustrations of the wedge product (We refer readers to (Lee, 2013) for more details).

Definition 6 (integral curve)**.** *Given a vector field X on M, an integral curve of X is a differentiable curve $\gamma : I \to M$, where $I \subseteq \mathbb{R}$ is an interval, whose velocity at each point is equal to the value of X at that point:*

$$\dot{\gamma}(t) = X_{\gamma(t)} \quad \text{for all } t \in I. \tag{41}$$

If $0 \in J$, the point $\gamma(0)$ is called the starting point of γ. From Picard's theorem (Hartman, 2002), we know that locally we always have an interval I on which the solution exists and is necessarily unique.

Definition 7 (exterior derivative and closed form)**.** *The exterior derivative is a linear operator that maps k-forms to $k+1$-forms. In local chart representation, we have that if ω is a k-form on M with the local representation as*

$$\omega = \omega_{a_1 \cdots a_k} \mathrm{d}x^{a_1} \wedge \cdots \wedge \mathrm{d}x^{a_k}. \tag{42}$$

Then, we have the exterior derivative

$$\begin{aligned} \mathrm{d}\omega &= \mathrm{d}\omega_{a_1 \cdots a_k} \wedge \mathrm{d}x^{a_1} \wedge \cdots \wedge \mathrm{d}x^{a_k} \\ &= \partial_b \omega_{a_1 \cdots a_k} \mathrm{d}x^b \wedge \mathrm{d}x^{a_1} \wedge \cdots \wedge \mathrm{d}x^{a_k}. \end{aligned} \tag{43}$$

A form ω is called closed *if $\mathrm{d}\omega = 0$.*

Theorem 2. *From (Rudin et al., 1976; Lee, 2013), we know that*

$$\mathrm{d} \circ \mathrm{d} \equiv 0, \tag{44}$$

which is a dual statement that the boundary of the boundary of a manifold is empty from Stokes' theorem.

G.2. Hamiltonian Vector Fields on Symplectic Cotangent Bundle

Definition 8 (symplectic vs. Riemannian)**.** *Let M be a smooth manifold.*

1. *symplectic form: A* 2-form *(so it is antisymmetric) ω is said to be a* symplectic form *on M if it is closed, i.e,*

$$\mathrm{d}\omega = 0,$$

and it is non-degenerate*, i.e,*

$$(\forall\, Y \in \Gamma(TM) : \omega(X, Y) = 0) \Rightarrow X = 0. \tag{45}$$

2. *metric tensor: A $(0,2)$* tensor field *g is said to be a* Riemannian metric *on M if it is* non-degenerate *and symmetric at each x, i.e.,*

$$g_x : T_xM \times T_xM \to \mathbb{R} \tag{46}$$
$$\text{s.t. } g_x(X_1, X_2) = g_x(X_2, X_1) \,\forall\, X_1, X_2 \in T_xM \tag{47}$$

Remark 1. *A manifold equipped with a symplectic form ω is called a* symplectic manifold*, while a manifold equipped with a metric tensor g is called a* pseudo-Riemannian manifold*. The Riemannian metric g is a $(0,2)$-tensor field measuring the norms of tangent vectors and the angles between them. To some extent, the "shape structure" of the manifold M is only available if we equipped M with a metric g.*

- *From the above definition, we know the symplectic form and the metric tensor are both nondegenerate bilinear $(0,2)$ tensor fields. One difference is the symplectic form is antisymmetric while the metric tensor is symmetric. At each point $x \in M$, if we use the local chart coordinate representation, the $(0,2)$ tensor can be represented as the following matrix multiplication*

$$\tilde{p}^\intercal W_x \tilde{p}$$

where $d \times d$ matrix W is symmetric for metric tensor and is skew-symmetric for symplectic form. $\tilde{p}^\intercal$ is the transpose of the velocity representation (which is a numerical vector) in a local chart.

- *Because of* non-degenerate*, on a symplectic manifold M, we can define an isomorphism between $\Gamma(TM)$ and $\Gamma(T^*M)$ by mapping a vector field $X \in \Gamma(TM)$ to a 1-form $\eta_V \in \Gamma(T^*M)$, where*

$$\eta_X(\cdot) \coloneqq \omega^2(\cdot, X) \tag{48}$$

Similarly, on a pseudo-Riemannian manifold, we can define an isomorphism between $\Gamma(TM)$ and $\Gamma(T^*M)$ by mapping a vector field $X \in \Gamma(TM)$ to a 1-form $\in \Gamma(T^*M)$, where

$$\alpha_g : TM \longrightarrow T^*M. \tag{49}$$

In a local chart coordinate representation, $\alpha_g = g_{ij}$ and its inverse $\alpha_g^{-1} = g^{ij}$ with $\sum_{j=1}^m g_{ij} g^{jk} = \delta_i^k$ and $\delta_i^k = 1$ iff $i = k$ and 0 otherwise. Note the components of the metric and the inverse metric are all taken in a given chart without explicitly mentioning them.

Definition 9 (Hamiltonian flow and orbit)**.** *For a general symplectic manifold M with a symplectic form ω^2, if we have $H \in \mathcal{C}^\infty(M)$, then $\mathrm{d}H$ is a differential 1-form on M. We define the vector field called the* ***Hamiltonian flow*** X_H ***associated to the Hamiltonian*** H*, which satisfies that*

$$\eta_{X_H}(\cdot) = \mathrm{d}H(\cdot).$$

The integral curves of are called ***Hamiltonian orbits of*** H*:*

$$\dot{\gamma}(t) = (X_H)_{\gamma(t)} \quad \text{for all } t \in I. \tag{50}$$

where $(X_H)_{\gamma(t)}$ is the tangent vector at $\gamma(t) \in M$.

Definition 10 (Poincaré 1-form and 2-form)**.** *On the cotangent bundle T^*M of a manifold M, we have a natural symplectic form, called the* Poincaré 1-form

$$\theta^1_{\text{Poincaré}} = p_i \mathrm{d}q^i. \tag{51}$$

Therefore, by the exterior derivative, we have the Poincaré 2-form

$$\omega^2_{\text{Poincaré}} = \mathrm{d}\theta^1 = \mathrm{d}(p_i \mathrm{d}q^i) = \sum_i \mathrm{d}p_i \wedge \mathrm{d}q^i \tag{52}$$

which is closed from (44). Therefore Poincaré 2-form *is a symplectic form on the cotangent bundle and cotangent bundles are the natural* ***phase spaces*** *of classical mechanics (De León & Rodrigues, 2011).*

For more general symplectic forms on the cotangent bundle, we can again use (44) to construct the closed 2-form from a 1-form which is potentially symplectic:

Corollary 1 ((Chen et al., 2021))**.** *According to (44), on the cotangent bundle T^*M of a manifold M, from a 1-form,*

$$\theta^1 = f_i \mathrm{d}q^i,$$

we derive a closed 2-form using the exterior derivative (43), and its local representation is given by the following

$$\omega^2 = \mathrm{d}\theta^1 = \mathrm{d}(f_i \mathrm{d}q^i) = \sum_{i<j} (\partial_i f_j - \partial_j f_i)\, \mathrm{d}p_i \wedge \mathrm{d}q^j$$

Remark 2. *Note, strictly speaking, we only can get the necessary "closed" condition for ω^2 to be potentially symplectic. However, it is enough for our use in our proposed framework.*

We now state one of the most fundamental results in symplectic geometry that links the general symplectic form to the special Poincaré 2-form.

Theorem 3 (Darboux (Lee, 2013))**.** *Let $(\tilde{M}, \tilde{\omega}^2)$ be a $2d$-dimensional symplectic manifold. For any $x \in \tilde{M}$, there are smooth coordinates $\left(\tilde{q}^1, \ldots, \tilde{q}^d, \tilde{p}^1, \ldots, \tilde{p}^n\right)$ centered at x in which ω^2 has the coordinate representation*

$$\tilde{\omega}^2 = \sum_{i=1}^{n} \mathrm{d}\tilde{q}^i \wedge \mathrm{d}\tilde{p}^i. \tag{53}$$

as a Poincaré 2-form, any coordinates satisfying (53) are called Darboux or symplectic coordinates.

Remark 3. *Therefore, for any symplectic form w^2 on the above cotangent bundle T^*M of M, we can always find a symplectic coordinate with the* Poincaré 2-form.

Corollary 2 ((Da Silva & Da Salva, 2008))**.** *According to (39), from the antisymmetric of the 2-forms, we have*

$$\omega^2(X_H, X_H) = 0 \tag{54}$$

which implies that hamiltonian vector fields preserve their hamiltonian functions H.

Remark 4. *In physics, hamiltonian functions are typically energy functions for a physical system. Corollary 2 indicates if the system updates its status according to the hamiltonian vector fields, the time-evolution of the system follows the law of conservation of energy.*

Definition 11 (Hamiltonian orbit generated from Poincaré 2-form (Lee, 2013))**.** *The Hamiltonian orbit generated by the Hamiltonian flow X_H on the cotangent bundle equipped with the Poincaré 2-form given in local coordinates (q, p) is given by*

$$\dot{q}^i = \frac{\partial H}{\partial p_i}, \quad \dot{p}_i = -\frac{\partial H}{\partial q^i}. \tag{55}$$

Definition 12 (cogeodesic orbits)**.** *If additionally M is equipped with a metric tensor g, i.e, if M is a pseudo-Riemannian manifold with metric g, and if we set the hamiltonian H on T^*M as*

$$H(q, p) = \frac{1}{2} g^{ij}(q) p_i p_j,$$

the Hamiltonian orbit generated from Poincaré 2-form is given by

$$\dot{q}^i = \frac{\partial H}{\partial p_i} = g^{ij} p_j, \quad \dot{p}_i = -\frac{\partial H}{\partial q^i} = -\frac{1}{2} \partial_i g^{jk} p_j p_k. \tag{56}$$

It is called the ***cogeodesic orbits of*** (M, g)*. The canonical projection of cogeodesic orbits under π (34) is called* geodesic *on the base manifold M which generalizes the notion of a "straight or shortest line" to manifold where the length is measured by the metric tensor.*