Persistence Enhanced Graph Neural Network

Qi Zhao* **Ze Ye†** **Chao Chen†** **Yusu Wang***
* Department of Computer Science and Engineering The Ohio State University
†Department of Biomedical Informatics Stony Brook University

Abstract

Local structural information can increase the adaptability of graph convolutional networks to large graphs with heterogeneous topology. Existing methods only use relatively simple topological information, such as node degrees. We present a novel approach leveraging advanced topological information, i.e., persistent homology, which measures the information flow efficiency at different parts of the graph. To fully exploit such structural information in real world graphs, we propose a new network architecture which learns to use persistent homology information to reweight messages passed between graph nodes during convolution. For node classification tasks, our network outperforms existing ones on a broad spectrum of graph benchmarks.

1 INTRODUCTION

Deep learning methods have achieved immense success in different domains such as computer vision and natural language processing (Goodfellow et al., 2016). While deep neural networks have shown strong performance on image or text data, their learning power is yet to be fully exploited on graph-structured data. At the same time, data with a latent graph structure is ubiquitous in modern data science. It is highly desirable to develop deep learning techniques that best suite graph structures, such as social network, knowledge network, brain connectivity network, etc.

Earlier works on Graph Neural Networks (GNNs) (Gori et al., 2005; Scarselli et al., 2009) use recursive networks. Those GNNs process the graph using a set of neurons, each corresponding to a node in the graph. The neurons update nodes representation and exchange information from linked neighbor nodes iteratively until reaching equilibrium.

Inspired by the power of convolutional networks on image and text data, different ideas have been proposed to implement the "convolution" on graph structures. There are two main directions, spectral convolutions and spatial convolutions. Spectral convolutional networks (Bruna et al., 2014; Defferrard et al., 2016) apply convolutions to the spectral domain or the frequency domain of the input graph. These methods tend to be efficient, but are highly graph-dependent.

In a more explicit manner, spatial convolutional networks implement convolutions on graphs. The feature representation of each node is iteratively updated by aggregating information from immediate neighbors (Hamilton et al., 2017; Xu et al., 2019) or a receptive field determined by special methods (Niepert et al., 2016). A transformation of the information from neighbors - either linear or non-linear - can be learned through training. The node representation information transferred between vertices are called *messages*. Veličković et al. (2018) used self-attention mechanism to further refine the messages based on local information, namely, features of source and target nodes.

In spatial convolutions, it is essential to have shared filter parameters across different parts of the graph. However, it has been observed that the filters should be adaptive to different local graph structures. In particular, node degrees have been used to reweight the messages or as additional features of node representations (Kipf and Welling, 2017; Monti et al., 2017). This way, messages relevant to hub nodes with high degrees will be different from messages between normal nodes. However, node degree is only the simplest graph structural property. There are much richer and advanced structural information that should be exploited in order to develop structure-adaptive convolutional filters.

In this paper, we propose a novel and principled approach to maximally leverage the structural information in spatial graph convolution. In particular, for

Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

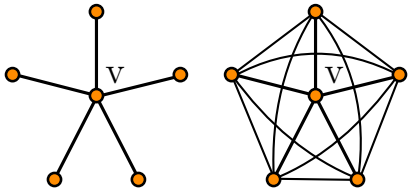

Figure 1: A node v is critical in a tree-structured neighborhood with no loops (left). But it is dispensable in a clique-structured neighborhood with 10 triangular loops containing v (right).

a node of the graph, we are interested in the *loopiness* of its neighborhood, i.e., how well its neighbors are inter-connected. Such structural property measures how critical the center node, v, is in information flow. At one extreme, if the neighborhood forms a loop-free tree, v is indispensable as any information between these neighbors has to pass v locally.[1] At another extreme, a clique neighborhood with N_v neighbors has $\binom{N_v}{2} = N_v(N_v-1)/2$ many triangular loops that contain v. In such case, any two neighbors can easily exchange information without going through v. See Figure 1 for illustrations. Such loopiness measure of a neighborhood should be fully exploited in graph learning, as it measures the information transmission efficiency of each node.

However, a direct cycle-counting within the neighborhood of v is insufficient. One major challenge is how to generalize to the cases when a graph is endowed with a metric, i.e., edges are annotated with different weights. These edge weights may come as part of the input information, e.g., frequency of communication between nodes in a social network, distance between nodes in a traffic network, etc. In other cases, these edge weights may be derived directly from the graph structural information. We need a robust 'cycle-counting' measure which should be stable to such graph metrics, i.e., does not change much when a metric is slightly perturbed.

To this end, we will use a principled mathematical tool, *persistent homology*, as a novel structural information measurement for graph convolutional networks. Persistent homology (Edelsbrunner et al., 2002; Edelsbrunner and Harer, 2010), as a modern adaption of the classic algebraic topology, can measure topological information carried in a metric space. In graph context, persistent homology not only counts the number of loops, but also measures the saliency of all loops in view of a given metric. The result of (Cohen-Steiner et al., 2007, 2010) shows that the structural information captured in persistent homology is stable with regard to certain perturbation of the underlying metric.

[1]The neighbors may communicate through other parts of the graph, but at a higher expense (longer route).

We propose *Persistence Enhanced Graph Network* (PEGN), a novel GNN to exploit the persistent homology information. We focus on the node classification task, while the method can be naturally generalized to graph classification. Our contribution is two-fold.

- We propose a novel network architecture for graphs, that uses persistent homology information in a data-driven manner. Based on the input persistent homology information of different neighborhoods, we train a separate network to reweight messages between nodes. This empowers the spatial graph convolution to be highly adaptive with regard to different local structures.
- We validate our proposed network on a broad spectrum of synthetic and real world graph datasets. Our method outperforms existing methods that only use node-feature-based attention or node degree information. This confirms the power of advanced structural information, i.e., persistent homology, in graph learning.

To the best of our knowledge, *we are the first to exploit advanced topological information beyond node degrees in spatial convolutional graph networks.* Note that spectral convolution methods, which run convolutions on the spectral or Fourier space, indirectly use advanced structural information. But for these methods, it is much harder to control the convolution with regard to different local structures as we do.

Related Work. Persistent homology plays a crucial role in topological data analysis, which extracts topological information from various geometric objects such as shapes, images or point clouds. Much progress has been made both on the theoretical front (e.g, (Edelsbrunner et al., 2002; Carlsson and Zomorodian, 2009; Chazal et al., 2009; Carlsson and de Silva, 2010; Chazal et al., 2016)) and the computational efficiency (e.g, (Sheehy, 2012; Bauer et al., 2014; Clément et al., 2014; Dey et al., 2016; Kerber and Schreiber, 2017; Bauer, 2016)). The extracted topological information has been used as powerful features in various contexts (Hofer et al., 2017; Adams et al., 2017; Carrire et al., 2017; Kusano et al., 2017). Advanced topological methods have been developed for image segmentation (Wu et al., 2017; Hu et al., 2019), clustering (Ni et al., 2017) and regularization of classifiers (Chen et al., 2019).

For graphs, persistent homology information has been used as a global structural signature for whole graph classification (Rieck et al., 2019; Li et al., 2017; Zhao and Wang, 2019). But no existing methods use such information as a local structural information to improve the adaptability of graph convolutional networks.

Outline. This paper is organized as follows. We first introduce the concepts and mathematical properties of persistent homology. We also show that the structural properties described as persistent homology are both stable and discriminative. Reassured by these theoretical guarantees, we propose the Persistence Enhanced Graph Network (PEGN). We present details of the network architecture. Finally, we show that our approach has achieved or matched the state-of-the-art across various graph benchmark datasets, in particular on larger and denser graphs.

2 PERSISTENT HOMOLOGY

We first provide a brief introduction of persistent homology and persistence diagrams. We refer the readers to (Edelsbrunner and Harer, 2010) for more details.

Suppose we are given a topological space X that we want to characterize. A *filtration of* X is a sequence of growing subsets: $X_1 \subseteq X_2 \subseteq \cdots \subseteq X_n = X$, which can be viewed as a specific way to inspect X. During the process of this inspection on X, sometimes a new topological feature (such as a component, a loop, a void, and so on) is created in X_i, and destroyed when entering X_j. *Persistent homology* can capture the *birth* and *death* of these topological features (quantified by homology classes). In particular, the *k-dimensional persistence diagram* $\mathrm{Dg}_k X$ consists of the diagonal[2] and a multi-set of *persistence points* in the (so-called "birth-death") plane, and each of the persistence point (b, d) records that some k-dimensional topological feature appears when entering X_b and disappears upon entering X_b. The *persistence* of this feature is its lifespan $|d-b|$. Overall, $\mathrm{Dg}X$ summarizes features of X w.r.t. the input filtration in a multi-scale manner.

There are many ways to build a filtration for a space X: different filtrations reflect different views of X and their corresponding persistence diagrams summarize different features of X. One common way to induce a filtration is via the superlevel-sets of a descriptor function $f : X \to \mathbb{R}$ defined on X. In particular, let $X_{\geq a} := \{x \in X | f(x) \geq a\}$ be the superlevel-set of f at a. Given an increasing sequence $a_1 > a_2 > \cdots > a_n$ of real values, the *superlevel-set filtration* on X w.r.t. f is

$$X_{\geq a_1} \subseteq X_{\geq a_2} \subseteq \cdots \subseteq X_{\geq a_n} = X \qquad (1)$$

Denote the persistence diagram induced by f as $\mathrm{Dg}f$. A persistence point $p = (a_i, a_j)$ indicates some topological features are created when entering $X_{\geq a_i}$ and destroyed upon entering $X_{\geq a_j}$. The persistence of the features is defined as its lifespan time $pers(p) = |a_j - a_i|$. Similarly, if X is swept bottom-up in increasing values, one gets the persistence diagram induced by sublevel-set filtration of X w.r.t. f.

Graph Setting. In our applications, our input will be graphs. Given a graph $G = (V, E)$, we can view it as a 1-dimensional simplicial complex. A (descriptor) function f defined on V or E will then induce a filtration as well as its persistence diagram summary. In particular, suppose $f : V \to \mathbb{R}$ is defined on the node set of G (e.g, the degree function). Then we can extend f to edges E of G by setting $f(u, v) = max\{f(u), f(v)\}$, and the sublevel-set at a is defined as $G_{\leq a} := \{\sigma \in V \cup E \mid f(\sigma) \leq a\}$. Similarly, if we are given $f : E \to \mathbb{R}$, then we can extend f to V by setting $f(u) = \min_{u \in e, e \in E} f(e)$. As we sweep G via the superlevel-set filtration of f, connected components in the swept subgraphs will be created and merged, and new cycles will be created. The formal events are encoded in the 0-dimensional persistence diagram $\mathrm{Dg}_0 f$. The 1-dimensional features (cycles), however, we note that cycles created will never be killed, as they are present in the total space $X = G$ (which is the last space in the sequence in Eqn (1)). To this end, we use the so-called *extended persistence* introduced in (Cohen-Steiner et al., 2009) which can record information of cycles.

An example of the persistence diagram induced by a function on graph is given in Figure 2. In this example, the descriptor function $f : V = \{u_1, \ldots, u_{10}\} \to \mathbb{R}$ is the shortest path distance function to the base point u_1; that is, for any u_i, $f(u_i) = d_G(u_i, u_1)$ where d_G denotes the shortest path metric on G. See Figure 2 (b), as the level decreases, the vertices enter the filtration in groups: $\{U_4\}$, $\{U_3, U_8, U_5\}$, $\{U2, U6, U7\}$ and $\{U_1\}$. Components are connected and cycles are created in this process. Points in the 0-D persistence diagram and the 1-D extended persistence diagram are shown in Figure 2 (c). For example, there are three independent loops in this graph, and a specific basis (the so-called "thinnest" system of loops) are captured in the 1-D extended persistence diagram (giving rise to three persistence points). u_4 gives rise to the 0-dim persistence point at $(3, 0)$, u_3, u_5, and u_8 induce the 0-dim persistence points at $(2, 1)$, while u_2, u_6, and u_7 induce the 0-dim persistence points at $(1, 0)$. Cycle L_1, L_2 and L_3 induce the 1-dim persistence points at $(3, 1)$, $(2, 1)$ and $(1, 0)$ respectively.

Metrics for Persistence Diagrams. Two most common ways to measure distances between persistence diagrams are the so-called bottleneck distance and the p-th Wasserstein distance. Both of these distances have been well studied in the literature, includ-

[2]Note that as is common in the literature, we also include the diagonal line into the persistence diagram $\mathrm{Dg}_k X$.

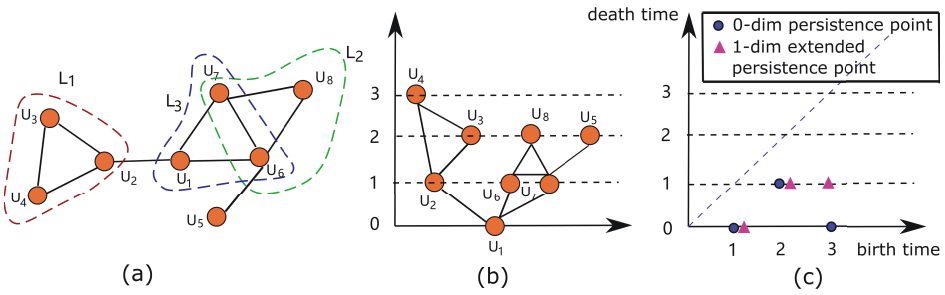

Figure 2: (a). An input weighted graph, where all edges have weight 1, other than edge (u_2, u_4) with weight $w(u_2, u_4) = 2$. (b). We re-plot the graph where the height of each node u_i equals to the descriptor function value $f(u_i)$. (c) shows the union of 0-D standard persistence diagram induced by the superlevel-set filtration (blue dots) and 1D extended persistence diagram (purple triangles).

ing stability results under these distances (e.g, (Cohen-Steiner et al., 2007, 2010; Chazal et al., 2016)) and efficient implementations (Kerber et al., 2017, 2018). In this paper, we focus on the Wasserstein distance, which we define below. Recall that the diagonal of the plane belongs to the persistence diagram.

Definition 1. *Let* D_1 *and* D_2 *be two persistence diagrams. The p-th Wasserstein distance between them is defined as:*

$$W_p(\mathrm{D}_1, \mathrm{D}_2)) = \inf_{\gamma: D_1 \to D_2} [\sum_{x \in D_1} ||x - \gamma(x)||^p]^{1/p} \quad (2)$$

where the infimum is over all bijections $\gamma : D_1 \to D_2$ *and the summation is over all points in* D_1*.*

Persistence Images. In recent years, there have been several approaches to further convert persistence diagrams to yet another representation, as a vector in a finite or infinite-dimensional Hilbert space, so as to facilitate downstream machine learning tasks; see e.g, (Bubenik, 2015). Below we introduce one such representations, called the persistence images (originally introduced in (Adams et al., 2017)), which we will use later in our persistence-enhanced GNN framework.

Set $T : \mathbb{R}^2 \to \mathbb{R}^2$ to be the linear transformation $T(x, y) = (x, y - x)$. Given a persistence diagram D, let $T(\mathrm{D})$ Denote the transformed diagram of a persistence diagram D as $T(\mathrm{D})$. Let $\phi_u : \mathbb{R}^2 \to \mathbb{R}$ be a differentiable distribution function whose mean locates at $u \in \mathbb{R}^2$: For example, in our implementation later, we will use the Gaussian distribution function (a common choice), where for any $z \in \mathbb{R}^2$, $\phi_u(z) = \frac{1}{2\pi\tau^2} e^{-\frac{||z-u||^2}{2\tau^2}}$.

Definition 2 (Persistence images)**.** *Let* $\alpha : \mathbb{R}^2 \to \mathbb{R}$ *be a non-negative weight function for the persistent plane* $\mathbb{R}^2$*. Given a persistence diagram* $\mathrm{Dg}X$*, its* persistence surface $\rho_{\mathrm{D}} : \mathbb{R}^2 \to \mathbb{R}$ *(w.r.t.* α*) is defined as: for any* $z \in \mathbb{R}^2$*,*

$$\rho_{\mathrm{D}}(z) = \sum_{u \in T(\mathrm{D})} \alpha(u)\phi_u(z). \quad (3)$$

The persistence image is a discretization of the persistence surface as follows. Set a fixed grid within a rectangle in the plane with a collection $\mathcal{P}$ *of N pixels. The* persistence image *for a persistence diagram* D *is* $\mathrm{PI}_{\mathrm{D}} = \{\mathrm{PI}_{\mathrm{D}}[p]\}_{p \in \mathcal{P}}$*, where* $\mathrm{PI}_{\mathrm{D}}[p] := \int \int_p \rho_{\mathrm{Dg}(X)} dydx$*.*

Note that PI_{D} can also be viewed as a vector in $\mathbb{R}^N$, and thus persistence images are naturally equipped with the L_2-distance in $\mathbb{R}^N$. Theorem 5 of (Adams et al., 2017) shows that the persistence images (under L_2-distance) are stable w.r.t. perturbations of the persistence diagrams under the 1-Wasserstein distance.

2.1 An Example on Stochastic Block Model

In this section, we will consider a graph $G = (V, E)$ generated by stochastic block model (SBM) with 2 communities as a toy example, to illustrate the information encoded in the persistence diagrams of subgraphs, and show that such local information can be used to differentiate nodes from different communities when they have different density. We consider persistence diagrams of local subgraphs, as they will be used later in our Persistence-enhanced graph neural networks. More explicitly, the following Theorem 1 indicates that the persistence diagrams of local subgraphs from different communities in SBM can be separated with a significantly large margin.

SBM Model. We assume that we are given a graph $G = (V, E)$ sampled from the following *stochastic block model*. Assume the vertices $V = C_1 \cup C_2$ are from two disjoint communities C_1 and C_2, with $n_1 = |C_1|$, $n_2 = |C_2|$ and $n = n_1 + n_2$. The edge set is sampled at random as follows: any two vertices $u, v \in C_i$ are connected by an edge with probability p_i, for $i = 1, 2$, while a pair of vertices $u \in C_1$ and $v \in C_2$ has an edge connecting them with probability $q < min\{p_1, p_2\}$. In other words, the in-cluster edge probability is p_1 and p_2, respectively, while the between-cluster edge prob-

ability is q. We say that graph $G = (V, E)$ is sampled from $\mathrm{SBM}(p_1, p_2, q, n_1, n_2)$. Equivalent, G is sampled from the $n \times n$ edge probability matrix P where $P_{uv} = p_i$ if both $u, v \in C_i$, and $P_{uv} = q$ otherwise.

Given such a graph G, we will use the following descriptor function $f : E \to \mathbb{R}$ to generate persistence diagrams. (Eldridge et al., 2016) proposed a simple edge-smoothing strategy to estimate the edge probability matrix P within L_∞ error. In particular, their algorithm outputs a new edge probability matrix $\widehat{P}$ such that with high probability $\mu = 1 - O((\frac{logn}{n})^{1/6})$, $|\widehat{P}_{uv} - P_{uv}| \leq \delta$ for all $u, v \in V$, where $\delta = o(1)$ is a quantity depending on n. For any edge $(u, v) \in E$, we simply set $f(u, v) = \widehat{P}_{uv}$.

Now given any node u from G, we consider its *1-hop neighborhood* subgraph $G_u = (V_u, E_u)$, where V_u contains u and all immediate neighbors of u, while E_u is the set of edges in G spanned by V_u. Let $f_u : E_u \to \mathbb{R}$ be the restriction of f over edge set E_u, and as described earlier, we can extend f_u to nodes V_u in G_u as well by setting $f_u(v) = \max_{(u,v)\in E_u} f(u, v)$. Let $\mathrm{Dg}_0 G_u$ and $\mathrm{Dg}_1 G_u$ denote the 0-D persistence diagram induced by the super-level set filtration of f_u, and 1-D extended persistence diagram induced by f_u, respectively. These local persistence diagrams $\mathrm{Dg} G_u$ provides a summary of the local neighborhood of u in G. We have the following theorem, whose proof can be found in the Supplement Material.

Theorem 1. *Let $G = (V, E)$ be a random graph sampled from $\mathrm{SBM}(p_1, p_2, q, n_1, n_2)$. Let $u \in C_1$ and $v \in C_2$, compute $\mathrm{Dg} G_u$ and $\mathrm{Dg} G_v$ as described above. Given any constant $\epsilon > 0$, the following two inequalities hold with high probability:*

$$\begin{aligned} &W_1(\mathrm{Dg}_0 G_u, \mathrm{Dg}_0 G_v) \geq \\ &\qquad c \cdot max\{n_1|p_1 - q - \epsilon|, n_2|p_2 - q - \epsilon|\} \\ &W_1(\mathrm{Dg}_1 G_u, \mathrm{Dg}_1 G_v) \geq \\ &\qquad c \cdot max\{n_1^2|p_1^3 - p_1 q^2 - 2\epsilon|, n_2^2|p_2^3 - p_2 q^2 - 2\epsilon|\} \end{aligned} \tag{4}$$

These calculations can be generalized to SBM with multiple (but fixed number of) communities.

Intuitively, Theorem 1 states that the persistence diagrams of local neighborhoods of nodes contain sufficient information to differentiating different communities when they have different densities. While this is a very simple example, it illustrates that persistence diagrams can contain stable and useful information. We remark that while for such simple models, it may be possible to use information such as node degree to differentiate nodes from different communities, in general, persistence encodes much richer information than node degree: for example, the extended persistence also encodes the cycle information.

3 PERSISTENCE ENHANCED GRAPH NETWORK

A Persistence Enhanced Graph Network (PEGN) is a spatial GNN. The convolution can be viewed as a message passing framework. Messages are passed between nodes in order to update their feature representation. After a fixed number of iterations, the feature representation of each node is used for classification or other tasks. Here we focus on a node classification task, in which node representations are used to predict labels of all nodes. In a graph classification task, these node representations can be aggregated to a graph representation and be fed into a classifier.

To effectively incorporate structural information into the framework, we propose a separate network, Persistence Image Network (PIN), converting persistent homology information into message reweighting vectors. These vectors are used to reweight messages and to effectively improve the graph convolution performance. See Figure 3 for the architecture of our network.

We first describe details of PEGN, such as how messages are computed and passed between nodes, and how the messages are reweighted. Next, we show how persistent homology information are converted into message reweighting vectors by PIN.

3.1 PEGN: Message Reweighting Graph Convolution

A graph neural network with L layers updates node feature representations for L times. Together with the input node features, we have $L + 1$ node feature representations, $H^\ell = [\vec{h_1^\ell}, \vec{h_2^\ell}, \cdots, \vec{h_N^\ell}]$, $\vec{h_n^\ell} \in \mathbb{R}^{d_\ell}$, $\ell = 0, \cdots, L$. Here N is the number of nodes in the graph. d_ℓ is the feature dimension for the ℓ-th layer representation. We denote by H^0 the input node features. H^L is the final layer feature representations and will be used for prediction.

A convolutional layer generates the ℓ-th layer representation using the $(\ell - 1)$-th layer representation. To compute the representation of node u, $\vec{h_u^\ell}$, we use the previous layer representations of u and its immediate neighbors, $\overline{\mathcal{N}}(u) = \{u\} \cup \mathcal{N}(u)$. These representations are transformed using a transformation matrix W^ℓ and are aggregated, formally,

$$\vec{h}_u^\ell = \sigma\left(\sum\nolimits_{v \in \overline{\mathcal{N}}(u)} W^\ell \vec{h}_v^{\ell-1}\right). \tag{5}$$

The transformation matrix W^ℓ for the ℓ-th layer is learned in training. The additional function σ is the nonlinear transformation, e.g., ReLU. The transformed representation $W^\ell \vec{h}_v^{\ell-1}$ is the message passed

from node v to node u. Note that all messages share a same transformation matrix W^ℓ.

To incorporate structural information, we introduce additional message reweighting vectors, $\tau^\ell_{v\to u}$. Reweighting vectors have the same length as the number of channels of a message. They are different for different edges, depending on the structural information associated with the edge. Formally, we write the representation updating equation as

$$\vec{h}^\ell_u = \sigma\left(\sum\nolimits_{v\in\overline{\mathcal{N}}(u)} \mathrm{diag}(\tau^\ell_{v\to u})W^\ell \vec{h}^{\ell-1}_v\right). \quad (6)$$

Node degree and self-attention mechanism have been applied to reweight messages. However, the structural information is much richer than simply node degree.

We propose to use persistent homology describing topology of local neighborhood graphs of u and v to generate the reweighting vector $\tau^\ell_{v\to u}$. It remains to explain how persistent homology information, namely, persistence diagrams can be transformed into the reweighting vector using our Persistence Image Network (PIN). See Figure 4 for the architecture. First, it converts a persistence diagram into a fixed-length vector, called the persistence image. Second, it converts persistence images of u and v into a reweighting vector using a multilayer perceptron (MLP).

3.2 Persistence Images from Subgraphs

We first describe how the demanded persistence images are generated. Consider the persistence images corresponding to nodes, a subgraph G_u can be picked around a node u by selecting a set of nodes closed to u. One method is to pick its $q-hop$ neighbourhood ($q \geq 1$) and another is to apply a random walk starting from u. The filtration function f defined over node set included in G_u is the minimum geodesic distance d_{ux} between a node x and u. Naturally, a persistence diagram and, to a step further, a persistence image can be computed as illustrated in Section 2. The condition of edges is a little bit more complicated. The subgraph G_{uv} around an edge (u, v) can be constructed from the union or intersection of G_u and G_v, and the filtration function f over node set V_{uv} of G_{uv} can be defined as

$$\begin{aligned} f(x) &= min\{d_{xu}, d_{xv}\}, \quad or \\ f(x) &= max\{d_{xu}, d_{xv}\} \end{aligned} \quad (7)$$

where $x \in G_{uv}$. Or more simply, $f(x)$ can be defined as d_{xu} or d_{xv} directly.

If we extract topological features from edge-wise subgraph G_{uv}, denote its corresponding persistence images as $PI(u, v)$. If we extract those from node-wise subgraph G_u, then the persistence image is denoted as $PI(u)$. Define the persistence information vector of any edge (u, v) as

$$\psi_{uv} = \begin{cases} PI(u,v) & \text{in edge-wise subgraph} \\ (PI(u), PI(v)) & \text{in node-wise subgraph} \end{cases} \quad (8)$$

where $(\cdot,\cdot)$ is the simple concatenation.

3.3 How Persistence Images Improve GNNs

Next we present how ψ_{uv} acts on the reweighting parameter τ^k_{uv} in graph convolution processes and promotes the performance of GNNs. See Figure 3 for an illustration of our framework.

Start from the setup that τ^k_{uv} is a scalar, there is a function $f^k : \mathbb{R}^l \to \mathbb{R}$ mapping ψ_{uv} to τ^k_{uv} where l is the dimension of persistence information vectors. It is desirable that f^k is a learnable parametric function with the idea of data-driven mechanism. As the multilayer perceptrons (MLPs) are universal function approximators shown by Cybenko's theorem (Cybenko, 1989) and easily carried into more complicated neural network models, we adopt MLPs to approximate the mapping function f^k. Notice the aggregations of messages are probably large arbitrarily, softmax functions are supposed to be applied for normalization after the MLPs processing. More exactly, denote the MLP and softmax function in the k_{th} layer as g^k and s^k respectively, the function approximating τ^k_{uv} is

$$\begin{aligned} f^k(\psi_{uv}) &= s^k(g^k(\psi_{uv})) \\ &= \frac{exp(g^k(\psi_{uv}))}{\sum_{x\in\mathcal{N}(u)} exp(g^k(\psi_{ux}))} \end{aligned} \quad (9)$$

Recall that in the k_{th} layer, the messages passing through edges are F^k dimensional, it is intuitive that a scalar form of τ^k_{uv} cannot exert all effects of persistence information vector ψ_{uv}. In other words, it is not necessary that reweighting parameters for different dimensions of features keep uniform. Instead a reweighting vector in F^k dimensions is cast to function $f^k : \mathbb{R}^l \to \mathbb{R}^{F^k}$.

$$f^k(\psi_{uv}) = \mathbf{S}^k(\mathbf{g}^k(\psi_{uv})) \quad (10)$$

If we view each dimension of ψ_{uv} as a channel, then $\mathbf{S}^k$ is channel-wise softmax function and $\mathbf{g}^k$ is an MLP outputting a F^k-dim vector. Notice that τ^k_{uv} is a F^k-dimensional vector rather than a scalar. Look back the messages aggregation and features update function (6), embed the persistence reweighting function into it,

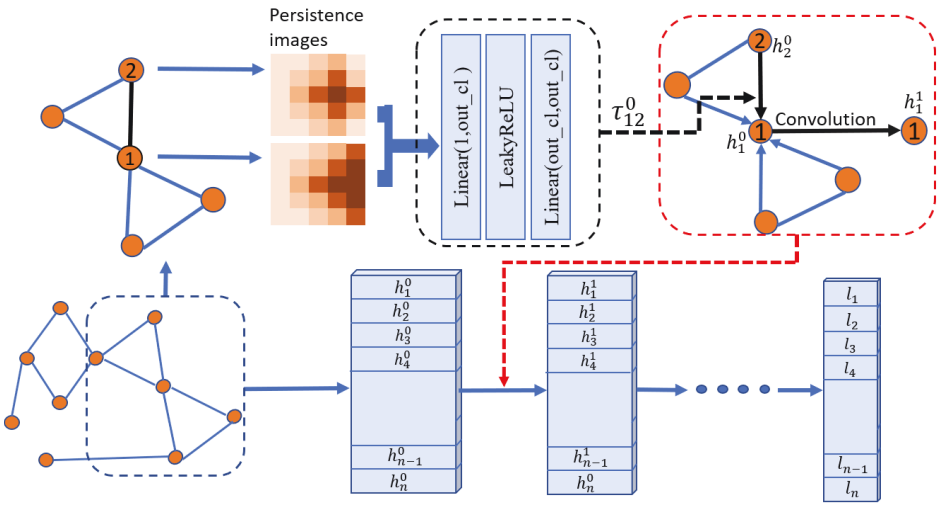

Figure 3: The framework of Persistence Enhanced Graph Network

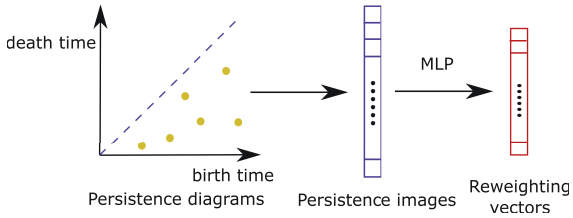

Figure 4: The framework of Persistence Image Network

the convolution layer of PEGN is

$$\begin{aligned}\vec{h}_u^k &= \sigma_{k-1}\Big(\sum_{v\in\bar{\mathcal{N}}(u)} diag(\tau_{uv}^{k-1})W^{k-1}\vec{h}_v^{k-1}\Big)\\ &= \sigma_{k-1}\Big(\sum_{v\in\bar{\mathcal{N}}(u)} diag(\mathbf{S}^{k-1}(\mathbf{g}^{k-1}(\psi_{uv})))W^{k-1}\vec{h}_v^{k-1}\Big)\end{aligned} \tag{11}$$

where $diag(\tau_{uv}^{k-1})$ is the diagonal matrix whose main diagonal entries are elements of τ_{uv}^{k-1}.

From layers above, we see the reweighting parameters τ_{uv}^k can even be a higher dimensional matrix, say $\mathbb{R}^{F^k} \times \mathbb{R}^{F^k}$ dimensions. However, the increasing of learnable parameters without significant mathematical meaning is not necessarily positive. In our experiments, the F^k-dimension vector formed τ_{uv}^k works well sufficiently.

4 EXPERIMENTS

We have compared our Persistence Enhanced Graph Network against a couple of popular and strong baselines across two categories of graph benckmark datasets. In this section, we introduce these datasets and baselines. We summarize our experimental setup and discuss results. We show that our model has achieved or matched the state-of-the-arts in these node classification benckmarks.

Datasets. We evaluate our graph network on three standard and widely used citation network benchmarks: Cora, Citeseer and Pubmed (Sen et al., 2008). We also use graph benchmarks such as Coauthor-CS, Coauthor-Physics, Amazon-Computers and Amazon-Photo (Shchur et al., 2018). The detailed introduction and statistics of these datasets are listed in the supplemental material. Pubmed, Coauthor and Amazon are larger and denser. Some of them consist of over 10,000 nodes and 200,000 edges with higher average node degrees. They are more challenging in node classification tasks.

Experimental setup. We compare our Persistence Enhanced Graph Network with a list of baselines: GCN (Kipf and Welling, 2017) reweighting messages according to node degrees, GraphSAGE (Hamilton et al., 2017) with mean aggregation of messages received within a sampled neighbourhood which works well in large graphs, MoNet (Monti et al., 2017), Graph U-Net (Gao and Ji, 2019), GAT (Veličković et al., 2018) adopting self-attention methods to reweight node features, and WLCN (Morris et al., 2019) also taking subgraph structures information. In addition, we provide the node classification performance for all datasets of MLP which does not incorporate any graph structure information.

In the practical experiments, as the graph is unweighted, we apply Ollivier's Ricci curvature (Ni et al., 2018) as the weight function in graphs, and construct subgraphs by picking $1-hop$ and $2-hop$ neighbourhoods around each node. Denote them as **PEGN-**

Table 1: Classification Accuracies on Benchmark Datasets

Method	Cora	Citeseer	PubMed	Coauthor CS	Coauthor Physics	Amazon Computer	Amazon Photo
MLP	58.2	59.1	70.0±2.1	88.3±0.7	88.9±1.1	44.9±5.8	69.6±3.8
MoNet	81.7	71.2	78.6±2.3	90.8±0.6	92.5±0.9	83.5±2.2	91.2±1.3
GraphSAGE	79.2	71.2	77.4±2.2	91.3±2.8	93.0±0.8	82.4±1.8	91.4±1.3
U-Net	82.5	72.0	78.9	92.7	94.0	86.0	91.9
WLCN	78.9	67.4	78.1	89.1	90.7	67.6	82.1
GCN	81.5±0.5	70.9±0.5	79.0±0.3	91.1±0.5	92.8±1.0	82.6±2.4	91.2±1.2
GAT	**83.0**±0.7	**72.5**±0.7	79.0±0.3	90.5±0.6	92.5±0.9	78.0±19.0	85.1±20.3
PEGN-RC-2	82.7±0.5	71.9±0.6	**79.4**±0.7	**92.9**±0.3	94.1±0.3	84.2±1	91.7±0.5
PEGN-RC-1	82.6±0.6	71.7±0.6	78.8±0.5	92.7±0.3	**94.2**±0.2	**86.3**±0.6	**92.5**±0.4

RC-1 and **PEGN-RC-2** respectively. All persistence images in the experiments are 25-dimension. A two-layer graph network model is evaluated. The first layer makes a linear transformation over the input node representations, and then reweights the output feature vectors by pairing with a vector computed from three layer MLP. The non-linear function σ_0 is an exponential linear unit function. The second layer has the same structure with the first layer except the dimension of output feature vector is the number of classes, namely the second layer is for classification.

The train-validation-test split policy is exactly the same as that of GCN and GAT in (Kipf and Welling, 2017; Veličković et al., 2018). Train graph networks with 20 nodes from each class, validate the algorithms on 500 nodes and test them on 1000 nodes. Models are initialized by Glorot initialization and the cross-entropy losses are minimized by Adam SGD optimizier with learning rate $r = 5e-3$. Additionally, we use L_2 regularization with $\lambda = 5e-4$ and early stopping based on the validation accuracy within 200 epochs.

The average classification accuracy and standard deviation are reported in Table 1 coming from 50 runs. We also collect the performances of baselines from (Monti et al., 2017; Shchur et al., 2018; Veličković et al., 2018).

Persistence Enhanced Graph Network is comparable with the state-of-the-art in the two small datasets, Cora and Citeseer, and outperforms baselines in other datasets, the larger and denser ones. Notice that although the performances of PEGN-RC-1 are slightly worse than PEGN-RC-2 on several datasets, it achieves higher average accuracy on dense Amazon graphs. Compared to GraphSAGE and U-Net aggregating information from multi-hop neighborhood and WLCN also taking subgraph structures information, it is fair to claim advanced topological structure information captured by persistence images indeed improve the performance of GNN in general, as the architecture of PEGN without reweighting mechanism is similar to that in GCN. We also provide the experiment results taking Jaccard index as the weight function in Supplement, which is similar to PEGN-RC. What's more, although multi-hop neighbourhood catches topological features from a wider range, it is not necessarily as distinguished as 1-hop neighbourhood in dense graphs. In particular, almost all of the 2-hop neighbourhood subgraphs in Amazon-Computers have more than 3000 nodes, which are considerably large parts of the entire graph. In other words, the topological features in 2-hop neighbourhood are not "local" sufficiently. Hence, multi-hop messages converge fast and are averaged out over the whole graph and produce similar node representations.

5 CONCLUSION

In this paper, we present Persistence Enhanced Graph Network, a novel architecture leveraging topological structure information with persistence images, a stable vectorized representation of persistence diagrams. Experiments show the scheme that passing messages are reweighted in accordance with topological features achieves or matches state-of-the-art across node classification benchmarks. One potential improvement comes from more elaborately designed subgraphs around nodes or edges and filtration functions. Moreover, extending Persistence Enhanced Graph Network to graph classification tasks would be an interesting research direction.

Acknowledgement. Zhao and Wang's research was partially supported by NSF grants CCF-1740761, CCF-1733798, and IIS-1815697. Ye and Chen's research was partially supported by NSF IIS-1909038, IIS-1855759, CCF-1855760.

References

Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: a stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18:218–252, 2017.

Ulrich Bauer. Ripser. `https://github.com/Ripser/ripser`, 2016.

Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat – persistent homology algorithms toolbox. In Hoon Hong and Chee Yap, editors, *Mathematical Software – ICMS 2014*, pages 137–143, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *International Conference on Learning Representations*, 2014.

Peter Bubenik. Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, 16(1):77–102, 2015.

G. Carlsson and A. Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, 2009.

Gunnar Carlsson and Vin de Silva. Zigzag persistence. *Foundations of Computational Mathematics*, 10(4): 367–405, 2010.

Mathieu Carrire, Marco Cuturi, and Steve Oudot. Sliced Wasserstein kernel for persistence diagrams. *International Conference on Machine Learning*, pages 664–673, 2017.

Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J. Guibas, and Steve Oudot. Proximity of persistence modules and their diagrams. In *Proc. 25th ACM Sympos. on Comput. Geom.*, pages 237–246, 2009.

Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The structure and stability of persistence modules*. SpringerBriefs in Mathematics. Springer, 2016.

Chao Chen, Xiuyan Ni, Qinxun Bai, and Yusu Wang. A topological regularizer for classifiers via persistent homology. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2573–2582, 2019.

Maria Clément, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The gudhi library: simplicial complexes and persistent homology. `http://gudhi.gforge.inria.fr/python/latest/index.html`, 2014.

David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Extending persistence using poincaré and lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, 2009.

David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Yuriy Mileyko. Lipschitz functions have Lp-stable persistence. *Foundations of computational mathematics*, 10(2):127–139, 2010.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

Tamal K. Dey, Dayu Shi, and Yusu Wang. Simba: An efficient tool for approximating Rips-filtration persistence via simplicial batch-collapse. In *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:16, 2016. ISBN 978-3-95977-015-6.

H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.*, 28:511–533, 2002.

Herbert Edelsbrunner and John Harer. *Computational Topology : an Introduction*. American Mathematical Society, 2010. ISBN 978-0-8218-4925-5.

Justin Eldridge, Mikhail Belkin, and Yusu Wang. Graphons, mergeons, and so on! In *Advances in Neural Information Processing Systems*, pages 2307–2315, 2016.

Hongyang Gao and Shuiwang Ji. Graph u-nets. In *International Conference on Machine Learning*, pages 2083–2092, 2019.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

Christoph Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. Deep learning with topological

signatures. In *Advances in Neural Information Processing Systems*, pages 1634–1644, 2017.

Xiaoling Hu, Fuxin Li, Dimitris Samaras, and Chao Chen. Topology-preserving deep image segmentation. In *Advances in Neural Information Processing Systems*, pages 5658–5669, 2019.

Michael Kerber and Hannah Schreiber. Barcodes of towers and a streaming algorithm for persistent homology. In *33rd International Symposium on Computational Geometry (SoCG 2017)*, page 57. Schloss Dagstuhl-Leibniz-Zentrum für Informatik GmbH, 2017.

Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *J. Exp. Algorithmics*, 22:1.4:1–1.4:20, September 2017. ISSN 1084-6654.

Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. HERA: software to compute distances for persistence diagrams. `https://bitbucket.org/grey_narn/hera`, 2018.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2017.

Genki Kusano, Kenji Fukumizu, and Yasuaki Hiraoka. Kernel method for persistence diagrams via kernel embedding and weight factor. *The Journal of Machine Learning Research*, 18(1):6947–6987, 2017.

Yanjie Li, Dingkang Wang, Giorgio A Ascoli, Partha Mitra, and Yusu Wang. Metrics for comparing neuronal tree shapes based on persistent homology. *PloS one*, 12(8):e0182184, 2017.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

Chien-Chun Ni, Yu-Yao Lin, Jie Gao, and Xianfeng Gu. Network alignment by discrete ollivier-ricci flow. In *International Symposium on Graph Drawing and Network Visualization*, pages 447–462. Springer, 2018.

Xiuyan Ni, Novi Quadrianto, Yusu Wang, and Chao Chen. Composing tree graphical models with persistent homology features for clustering mixed-type data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2622–2631. JMLR. org, 2017.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016.

Bastian Rieck, Christian Bock, and Karsten Borgwardt. A persistent weisfeiler-lehman procedure for graph classification. In *International Conference on Machine Learning*, pages 5448–5458, 2019.

F Scarselli, M Gori, Ah Chung Tsoi, M Hagenbuchner, and G Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 1(20):61–80, 2009.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

D. Sheehy. Linear-size approximations to the Vietoris-Rips filtration. In *Proc. 28th. Annu. Sympos. Comput. Geom.*, pages 239–248, 2012.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representations*, 2018.

Pengxiang Wu, Chao Chen, Yusu Wang, Shaoting Zhang, Changhe Yuan, Zhen Qian, Dimitris Metaxas, and Leon Axel. Optimal topological cycles and their application in cardiac trabeculae restoration. In *International Conference on Information Processing in Medical Imaging*, pages 80–92. Springer, 2017.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *International Conference on Learning Representations*, 2019.

Qi Zhao and Yusu Wang. Learning metrics for persistence-based summaries and applications for graph classification. In *Advances in Neural Information Processing Systems*, pages 9855–9866, 2019.