Graph Mixup on Approximate Gromov–Wasserstein Geodesics

Zhichen Zeng [1] **Ruizhong Qiu** [1] **Zhe Xu** [1] **Zhining Liu** [1] **Yuchen Yan** [1]
Tianxin Wei [1] **Lei Ying** [2] **Jingrui He** [1] **Hanghang Tong** [1]

Abstract

Mixup, which generates synthetic training samples on the data manifold, has been shown to be highly effective in augmenting Euclidean data. However, finding a proper data manifold for graph data is non-trivial, as graphs are non-Euclidean data in disparate spaces. Though efforts have been made, most of the existing graph mixup methods neglect the intrinsic geodesic guarantee, thereby generating inconsistent sample-label pairs. To address this issue, we propose GeoMix to mixup graphs on the Gromov-Wasserstein (GW) geodesics. A joint space over input graphs is first defined based on the GW distance, and graphs are then transformed into the GW space through equivalence-preserving transformations. We further show that the linear interpolation of the transformed graph pairs defines a geodesic connecting the original pairs on the GW manifold, hence ensuring the consistency between generated samples and labels. An accelerated mixup algorithm on the approximate low-dimensional GW manifold is further proposed. Extensive experiments show that the proposed GeoMix promotes the generalization and robustness of GNN models.

1. Introduction

In the era of big data and AI, graphs are ubiquitous in various domains carrying rich information. Graph Neural Networks (GNNs) have achieved remarkable success in enormous graph learning tasks, including graph classification (Xu et al., 2018), node classification (Kipf & Welling, 2017; Xu et al., 2022a), link prediction (Zhang & Chen, 2018; Yan et al., 2024c;b), and many more. For (semi-)supervised learning tasks, the superior performance of GNNs largely depends on the training graphs, which are often noisy and scarce, inevitably inducing model over-fitting (Ding et al., 2022). To address this issue, graph data augmentation has been adopted for better model generalization by generating synthetic training graphs.

Mixup (Zhang et al., 2018) has achieved great success in computer vision (Zhang et al., 2018; Verma et al., 2019) and natural language processing (Guo et al., 2019a; Guo, 2020; Chen et al., 2020) in improving model generalization and robustness. The general idea is to linearly interpolate sample pairs on the data manifold, which defines a geodesic between sample pairs in the Euclidean space. However, mixup on graph data is non-trivial due to three key challenges. First (*space disparity*), as different graphs lie in disparate spaces, it is a prerequisite to find a joint space of graphs, i.e., space of spaces (Sturm, 2012), for mixup. Second (*non-Euclidean data*), even with a joint graph space, it is still challenging to interpolate graph pairs as graphs vary in sizes and are often not well-aligned (Han et al., 2022). Third (*sample–label consistency*), it is essential to ensure the consistency between mixup samples and labels, i.e., similar samples should share similar labels (Guo et al., 2019b; Kim et al., 2020; Liu et al., 2023a), but it remains unknown how to define and ensure such consistency for graph data.

To address the space disparity and non-Euclidean issues, existing methods have been focusing on design practical interpolation so that mixup in the graph space can be implemented in a similar way as mixup in the Euclidean space. For example, (Wang et al., 2021; Verma et al., 2021) transform graphs into vector embeddings and (Han et al., 2022; Guo & Mao, 2021; Ling et al., 2023) transform graph into well-aligned pairs, and mixup is performed between the transformed counterparts. A fundamental assumption behind these approaches is that the original graph and the transformed graph are equivalent so that the mixup samples between transformed graphs correspond to the mixup samples between the original graphs. Nevertheless, most of the transformations are not *equivalence-preserving*, thereby facing the risk of generating inconsistent sample–label pairs that *hurt*, as opposed to benefit, model training. Therefore, we take a step further and ask:

What kind of interpolation guarantees sample–label consistency in graph mixup from the geodesic perspective?

[1]University of Illinois Urbana-Champaign [2]University of Michigan, Ann Arbor. Correspondence to: Hanghang Tong <htong@illinois.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

Contributions. In this paper, we address the sample–label consistency issue in graph mixup and propose a novel algorithm named GEOMIX to interpolate graphs on the Gromov–Wasserstein (GW) geodesics. A comparison between the proposed GEOMIX and existing practical graph interpolation methods is shown in Figure 1. We first define sample–label consistency based on the GW distance, and show that interpolating graphs on the GW geodesics guarantees such consistency. We then construct a GW space (Mémoli, 2011; Sturm, 2012) as a unified graph space, and employ equivalence-preserving transformations (EPT) to transform graphs to their equivalent counterparts that are well-aligned in the GW space. We theoretically prove that the linear interpolation of the transformed graph pairs defines a geodesic connecting the original graph pairs in the GW space, hence ensuring the consistency between mixup samples and labels. For faster computation, an accelerated algorithm is introduced to mixup graphs on the approximate GW geodesic in the low-dimensional GW space, achieving quadratic time complexity w.r.t. the number of nodes in the input graph. Extensive experiments on real-world graphs show that GEOMIX improves GNN generalization and robustness, achieving up to 6.6% outperformance compared with the state-of-the-art graph mixup methods.

The rest of the paper is organized as follows. Section 2 introduces and analyzes the background knowledge on mixup and GW geometry. Section 3 introduces our proposed GEOMIX followed by experiments in Section 4. Related works and conclusions are given in Sections 5 and 6, respectively.

2. Preliminaries

2.1. Notations

We use bold uppercase letters for matrices (e.g., $\boldsymbol{A}$), bold lowercase letters for vectors (e.g., $\boldsymbol{s}$), calligraphic letters for sets (e.g., $\mathcal{G}$), and lowercase letters for scalars (e.g., α). The element (i, j) of a matrix $\boldsymbol{A}$ is denoted as $\boldsymbol{A}(i, j)$. The transpose of $\boldsymbol{A}$ is denoted by the superscript $\intercal$. The simplex histogram with n bins is denoted as $\Delta_n = \{\boldsymbol{\mu} \in \mathbb{R}_n^+ | \sum_{i=1}^n \boldsymbol{\mu}(i) = 1\}$. A coupling is denoted as $\Pi(\cdot, \cdot)$, and the inner product is denoted as $\langle \cdot, \cdot \rangle$. A graph is represented as a tuple $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{\mu})$ where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix, and $\boldsymbol{\mu} \in \Delta_n$ is the node weight. Without any prior knowledge on nodes, we default the node weight as uniform $\boldsymbol{\mu} = \frac{\mathbf{1}_n}{n}$. We denote the node set of a graph $\mathcal{G}$ as $\mathcal{V}(\mathcal{G})$.

2.2. Graph Mixup

Mixup is a simple yet effective augmentation method for Euclidean data like images. Given a pair of samples $\boldsymbol{x}_1, \boldsymbol{x}_2$ with their labels $\boldsymbol{y}_1, \boldsymbol{y}_2$, mixup generates synthetic samples by linearly interpolating the sample pairs and their labels:

$$\boldsymbol{x}_{(\lambda)} = (1 - \lambda)\boldsymbol{x}_1 + \lambda \boldsymbol{x}_2, \boldsymbol{y}_{(\lambda)} = (1 - \lambda)\boldsymbol{y}_1 + \lambda \boldsymbol{y}_2. \quad (1)$$

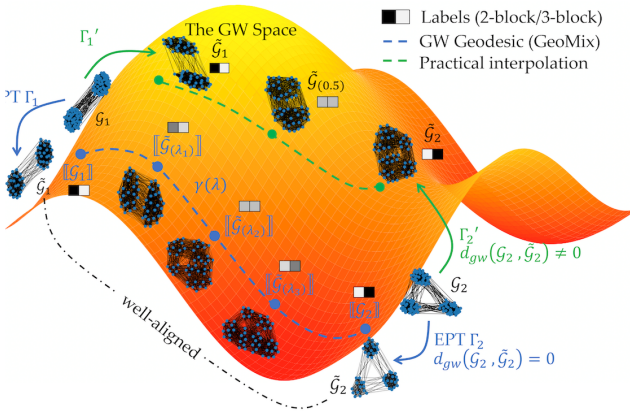


Figure 1. A comparison between GEOMIX and practical interpolation. GEOMIX (blue line) adopts equivalence-preserving transformations (EPT) to transform $\mathcal{G}_1, \mathcal{G}_2$ into two well-aligned $\tilde{\mathcal{G}}_1 \in [\![\mathcal{G}_1]\!], \tilde{\mathcal{G}}_2 \in [\![\mathcal{G}_2]\!]$, and then mixup on the GW geodesics $\gamma(\lambda) = [\![\tilde{\mathcal{G}}_{(\lambda)}]\!]$, hence ensuring sample–label consistency. While practical interpolation (green line) often ignore the equivalence between original and transformed graphs, resulting in inconsistent sample–label pairs, e.g., $\tilde{\mathcal{G}}_{(0.5)}$ is a 2-block graph but is labelled as half 2-block half 3-block. Best viewed in color.

In the graph setting, in order to interpolate graphs with different sizes that are not well-aligned, most of the existing methods follow a practical interpolation approach by interpolating the well-aligned transformed graphs, that is

$$\tilde{\mathcal{G}}_{(\lambda)} = (1-\lambda)\Gamma_1(\mathcal{G}_1) + \lambda \Gamma_2(\mathcal{G}_2), \tilde{\boldsymbol{y}}_{(\lambda)} = (1-\lambda)\boldsymbol{y}_1 + \lambda \boldsymbol{y}_2,$$

where Γ_1, Γ_2 are graph transformations. For example, in G-mixup (Han et al., 2022), Γ_1, Γ_2 map original graphs to the well-aligned graphons; in M-mixup (Wang et al., 2021), Γ_1, Γ_2 are neural encoders encoding graphs into the same hidden space; in S-mixup (Ling et al., 2023), Γ_1 corresponds to the soft alignment matrix between $\mathcal{G}_1$ and $\mathcal{G}_2$.

2.3. The Gromov–Wasserstein Space

Gromov–Wasserstein (GW) distance. The GW distance is a powerful approach to measure the distance between two spaces. We formally define the GW distance in the graph context as follows. Given two graphs $\mathcal{G}_1 = (\boldsymbol{A}_1, \boldsymbol{\mu}_1)$ and $\mathcal{G}_2 = (\boldsymbol{A}_2, \boldsymbol{\mu}_2)$, the p-GW distance between $\mathcal{G}_1$ and $\mathcal{G}_2$ is defined as (Mémoli, 2011):

$$d_{\mathrm{GW}}(\mathcal{G}_1, \mathcal{G}_2) = \min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)} \left(\varepsilon_{\boldsymbol{A}_1, \boldsymbol{A}_2}(\boldsymbol{T})\right)^{1/p}, \quad (2)$$

where p is the order of the GW distance, and

$$\varepsilon_{\boldsymbol{A}_1, \boldsymbol{A}_2}(\boldsymbol{T}) = \sum_{i,j,k,l} |\boldsymbol{A}_1(i, j) - \boldsymbol{A}_2(k, l)|^p \boldsymbol{T}(i, k)\boldsymbol{T}(j, l).$$

Intuitively, the GW distance yields an optimal matching $\boldsymbol{T}$ between two graphs regarding the connectivity structure

(i.e., $\boldsymbol{A}_1(i,j)-\boldsymbol{A}_2(k,l)$), together with a distance measured by the sum of node pair connectivity distances weighted by $\boldsymbol{T}$. In this paper, we adopt the 2-GW distance which has the following matrix form (Peyré et al., 2016):

$$\varepsilon_{\boldsymbol{A}_1,\boldsymbol{A}_2}(\boldsymbol{T})=\mathrm{Tr}(\boldsymbol{A}_1^2\boldsymbol{\mu}_1\boldsymbol{\mu}_1^\intercal)+\mathrm{Tr}(\boldsymbol{\mu}_2\boldsymbol{\mu}_2^\intercal\boldsymbol{A}_2^{2^\intercal})-2\mathrm{Tr}(\boldsymbol{A}_1\boldsymbol{T}\boldsymbol{A}_2^\intercal\boldsymbol{T}^\intercal).$$

GW geodesics. Note that the GW distance is only a *pseudometric* but not a *metric*. To define geodesics, we employ a standard procedure (Howes, 1995) to identify an induced metric d^*_{GW}. Define an equivalence relation $\sim$ over graphs: $\mathcal{G}_1 \sim \mathcal{G}_2$ iff $d_{\mathrm{GW}}(\mathcal{G}_1,\mathcal{G}_2)=0$. Let $[\![\mathcal{G}]\!] := \{\mathcal{G}' : \mathcal{G}' \sim \mathcal{G}\}$ denote the equivalence class of graph $\mathcal{G}$ w.r.t. $\sim$, and let $\mathfrak{G}$ denote the space of equivalence classes w.r.t. $\sim$. Then, the *induced metric* $d^*_{\mathrm{GW}} : \mathfrak{G}\times\mathfrak{G}\to\mathbb{R}_{\geq 0}$ is defined by

$$d^*_{\mathrm{GW}}([\![\mathcal{G}_1]\!],[\![\mathcal{G}_2]\!]) := d_{\mathrm{GW}}(\mathcal{G}_1,\mathcal{G}_2).$$

Note that d^*_{GW} is well defined as $d_{\mathrm{GW}}(\mathcal{G}_1,\mathcal{G}_2)$ is the same for any $\mathcal{G}_1 \in [\![\mathcal{G}_1]\!], \mathcal{G}_2 \in [\![\mathcal{G}_2]\!]$ (Howes, 1995). Based on the notion of equivalence class, an equivalence-preserving graph transformation is defined as follows:

Definition 2.1. *Equivalence-Preserving Transformation.* Given a graph space $\mathfrak{G}$, a graph transformation $\Gamma : \mathfrak{G}\to\mathfrak{G}$ is equivalence-preserving if any graph $\mathcal{G}\in\mathfrak{G}$ is equivalent to its transformed graph $\Gamma(\mathcal{G})$, i.e., $\mathcal{G}\sim\Gamma(\mathcal{G}), \forall\mathcal{G}\in\mathfrak{G}$.

Definition 2.2. *Gromov–Wasserstein Geodesics.* A curve $\gamma : [0,1]\to\mathfrak{G}$ is called a *GW geodesic* from $[\![\mathcal{G}_1]\!]$ to $[\![\mathcal{G}_2]\!]$ iff $\gamma(0)=[\![\mathcal{G}_1]\!]$, $\gamma(1)=[\![\mathcal{G}_2]\!]$, and for every $\lambda_1,\lambda_2\in[0,1]$,

$$d^*_{\mathrm{GW}}(\gamma(\lambda_1),\gamma(\lambda_2)) = |\lambda_1-\lambda_2|\cdot d^*_{\mathrm{GW}}([\![\mathcal{G}_1]\!],[\![\mathcal{G}_2]\!]). \quad (3)$$

2.4. Geodesic Graph Mixup

It is essential to ensure the consistency between mixup samples and labels to avoid suspicious supervision signals that may mislead the model. Intuitively, if two graphs are similar to each other, we expect them to share similar labels. Based on the GW distance, we define sample–label consistency as follows.

Definition 2.3. *Sample–Label Consistency*. Given two graphs $\mathcal{G}_1,\mathcal{G}_2$ with labels $\boldsymbol{y}_1,\boldsymbol{y}_2$, the mixup samples $\tilde{\mathcal{G}}_{(\lambda)}$ and labels $\tilde{\boldsymbol{y}}_{(\lambda)}$ are consistent iff for every $\lambda\in[0,1]$,

$$d_{\mathrm{GW}}(\tilde{\mathcal{G}}_{(\lambda)},\mathcal{G}_1)\|\tilde{\boldsymbol{y}}_{(\lambda)}-\boldsymbol{y}_2\| = d_{\mathrm{GW}}(\tilde{\mathcal{G}}_{(\lambda)},\mathcal{G}_2)\|\tilde{\boldsymbol{y}}_{(\lambda)}-\boldsymbol{y}_1\|.$$

When the transformations ignore the equivalence-preserving property, as most of the existing graph mixup methods do, the sample–label consistency is clearly violated as $d(\tilde{\mathcal{G}}_{(0)},\mathcal{G}_1) > 0 = \|\tilde{\boldsymbol{y}}_{(0)}-\boldsymbol{y}_1\|$. The following proposition states that samples on the GW geodesics satisfy the sample–label consistency.

Proposition 2.4. *The mixup samples $\tilde{\mathcal{G}}_{(\lambda)}$ and labels $\tilde{\boldsymbol{y}}_{(\lambda)}$ are consistent if $\tilde{\mathcal{G}}_{(\lambda)}$ are on the geodesic connecting original samples $\mathcal{G}_1,\mathcal{G}_2$.*

It is easy to verify the sample–label consistency for samples on the GW geodesics as follows:

$$\begin{array}{c} d_{\mathrm{GW}}(\tilde{\mathcal{G}}_{(\lambda)},\mathcal{G}_1)=\lambda d_{\mathrm{GW}}(\mathcal{G}_1,\mathcal{G}_2),\, d_{\mathrm{GW}}(\tilde{\mathcal{G}}_{(\lambda)},\mathcal{G}_2)=(1-\lambda)d_{\mathrm{GW}}(\mathcal{G}_1,\mathcal{G}_2) \\ \|\tilde{\boldsymbol{y}}_{(\lambda)}-\boldsymbol{y}_1\|=\lambda\|\boldsymbol{y}_1-\boldsymbol{y}_2\|,\, \|\tilde{\boldsymbol{y}}_{(\lambda)}-\boldsymbol{y}_2\|=(1-\lambda)\|\boldsymbol{y}_1-\boldsymbol{y}_2\| \end{array}.$$

Therefore, we study the geodesic graph mixup problem as follows.

Definition 2.5. *Geodesic Graph Mixup*. Given two graphs $\mathcal{G}_1,\mathcal{G}_2$, the geodesic graph mixup problem seeks for a GW geodesic $\gamma(\lambda)=[\![\tilde{\mathcal{G}}_{(\lambda)}]\!]$ connecting $\mathcal{G}_1,\mathcal{G}_2$ such that Eq. (3) is satisfied.

3. Methodology

In this section, we present and analyze our proposed GEOMIX. To generalize the Euclidean mixup to graphs, we first propose to mixup graphs on the exact GW geodesics in Section 3.1. To avoid the high dimensionality of exact GW geodesics, an accelerated algorithm on the approximated GW geodesics is introduced in Section 3.2.

3.1. Mixup on Exact GW Geodesics

Most of the existing graph mixup methods generalize mixup from the practical interpolation perspectives, which may induce inconsistency between the mixup samples and labels. Instead, we derive a more principled generalization of mixup from a *geodesic* perspective. We view Euclidean mixup in Eq. (1) as the *Euclidean geodesic* connecting $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. From this perspective, a natural generalization for graphs is the *GW geodesic* connecting two graphs. Thus, it suffices to find appropriate transformations Γ_1,Γ_2 such that $[\![(1-\lambda)\Gamma_1(\mathcal{G}_1)+\lambda\Gamma_2(\mathcal{G}_2)]\!]$ is the GW geodesic connecting two equivalence classes $[\![\mathcal{G}_1]\!],[\![\mathcal{G}_2]\!]$.

Thanks to the advancement in the optimal transport theory, the concept of geodesic has been generalized to the GW space to handle non-Euclidean data (Mémoli, 2011). According to Theorem 3.1 in (Sturm, 2012), given two graphs $\mathcal{G}_1,\mathcal{G}_2$, the 2-GW geodesic connecting $[\![\mathcal{G}_1]\!],[\![\mathcal{G}_2]\!]\in\mathfrak{G}$ is of the form $([\![\tilde{\mathcal{G}}_{(\lambda)}]\!])_{0\leq\lambda\leq 1}$ where $\tilde{\mathcal{G}}_{(\lambda)}=(\tilde{\boldsymbol{A}}_{(\lambda)},\tilde{\boldsymbol{\mu}}_{(\lambda)})$ is a graph with $\mathcal{V}(\tilde{\mathcal{G}}_{(\lambda)})=\mathcal{V}(\mathcal{G}_1)\times\mathcal{V}(\mathcal{G}_2)$ defined by Eq. (4).

$$\begin{aligned} \tilde{\boldsymbol{A}}_{(\lambda)} &:= (1-\lambda)\boldsymbol{A}_1\otimes\mathbf{1}_{n_2\times n_2}+\lambda\mathbf{1}_{n_1\times n_1}\otimes\boldsymbol{A}_2 \\ \tilde{\boldsymbol{\mu}}_{(\lambda)} &= \mathrm{vec}(\mathrm{OT}(\mathcal{G}_1,\mathcal{G}_2)) \end{aligned} \quad (4)$$

where $\mathrm{OT}(\mathcal{G}_1,\mathcal{G}_2)$ is the optimal coupling between $\mathcal{G}_1,\mathcal{G}_2$ given by the GW distance in Eq. (2). This theorem inspires us to consider two linear transformations $\boldsymbol{P}_1\in\mathbb{R}^{n_1\times n_1n_2}, \boldsymbol{P}_2\in\mathbb{R}^{n_2\times n_1n_2}$ transforming $\mathcal{G}_1,\mathcal{G}_2$ to $\tilde{\mathcal{G}}_1=$

$(\tilde{\boldsymbol{A}}_1, \tilde{\boldsymbol{\mu}}_1), \tilde{\mathcal{G}}_2 = (\tilde{\boldsymbol{A}}_2, \tilde{\boldsymbol{\mu}}_2)$ as follows

$$\begin{cases} \tilde{\boldsymbol{A}}_1 = \boldsymbol{P}_1^\top \boldsymbol{A}_1 \boldsymbol{P}_1, \tilde{\boldsymbol{\mu}}_1 = \mathrm{vec}(\mathrm{OT}(\mathcal{G}_1, \mathcal{G}_2)) \\ \tilde{\boldsymbol{A}}_2 = \boldsymbol{P}_2^\top \boldsymbol{A}_2 \boldsymbol{P}_2, \tilde{\boldsymbol{\mu}}_2 = \mathrm{vec}(\mathrm{OT}(\mathcal{G}_1, \mathcal{G}_2)) \end{cases} \quad \text{where} \begin{cases} \boldsymbol{P}_1 = \boldsymbol{I}_{n_1} \otimes \boldsymbol{1}_{1\times n_2} \\ \boldsymbol{P}_2 = \boldsymbol{1}_{1\times n_1} \otimes \boldsymbol{I}_{n_2} \end{cases}. \tag{5}$$

Each column of $\boldsymbol{P}_1, \boldsymbol{P}_2$ is an one-hot vector mapping nodes in $\mathcal{V}(\mathcal{G}_1)$ and $\mathcal{V}(\mathcal{G}_2)$ to $\mathcal{V}(\mathcal{G}_1) \times \mathcal{V}(\mathcal{G}_2)$, and the transformation in Eq. (5) is essentially a Kronecker product between the adjacency matrix and an all-one matrix, i.e., $\tilde{\boldsymbol{A}}_1 = \boldsymbol{A}_1 \otimes \boldsymbol{1}_{n_2\times n_2}, \tilde{\boldsymbol{A}}_2 = \boldsymbol{1}_{n_1\times n_1} \otimes \boldsymbol{A}_2$. Therefore, the transformed graphs $\tilde{\mathcal{G}}_1$ and $\tilde{\mathcal{G}}_2$ are in the same graph space, i.e., of the same size and well-aligned. On top of this, the mixup graph $\tilde{\mathcal{G}}_{(\lambda)} = (\tilde{\boldsymbol{A}}_{(\lambda)}, \tilde{\boldsymbol{\mu}}_{(\lambda)})$ and its label $\tilde{\boldsymbol{y}}_{(\lambda)}$ can be further defined as:

$$\begin{aligned} \tilde{\boldsymbol{A}}_{(\lambda)} &= (1-\lambda)\tilde{\boldsymbol{A}}_1 + \lambda \tilde{\boldsymbol{A}}_2, \\ \tilde{\boldsymbol{y}}_{(\lambda)} &= (1-\lambda)\boldsymbol{y}_1 + \lambda \boldsymbol{y}_2. \end{aligned} \tag{6}$$

The following theorem shows that the mixup graphs $\tilde{\mathcal{G}}_{(\lambda)}$ precisely defines the GW geodesic connecting $[\![\mathcal{G}_1]\!], [\![\mathcal{G}_2]\!]$.

Theorem 3.1. *Given two graphs $\mathcal{G}_1, \mathcal{G}_2$, the transformed graphs $\tilde{\mathcal{G}}_1, \tilde{\mathcal{G}}_2$ in Eq. (5) are in the equivalent class of $\mathcal{G}_1, \mathcal{G}_2$, respectively, that is*

$$[\![\tilde{\mathcal{G}}_1]\!] = [\![\mathcal{G}_1]\!], [\![\tilde{\mathcal{G}}_2]\!] = [\![\mathcal{G}_2]\!]$$

Furthermore, the curve $\{[\![\tilde{\mathcal{G}}_{(\lambda)}]\!]\}_{0\le\lambda\le 1}$ is a GW geodesic connecting $[\![\mathcal{G}_1]\!]$ and $[\![\mathcal{G}_2]\!]$, which means that for every $\lambda_1, \lambda_2 \in [0,1]$,

$$d^*_{\mathrm{GW}}([\![\tilde{\mathcal{G}}_{(\lambda_1)}]\!], [\![\tilde{\mathcal{G}}_{(\lambda_2)}]\!]) = |\lambda_1 - \lambda_2| \cdot d^*_{\mathrm{GW}}([\![\mathcal{G}_1]\!], [\![\mathcal{G}_2]\!]).$$

Proof is provided in Appendix A. This theorem, to the best of our knowledge, is the first to provide theoretical guarantee on the geodesic property for graph mixup.

3.2. Accelerating Mixup on Approximate GW Geodesics

Though achieving exact GW geodesics, the high dimensionality of $\tilde{\mathcal{G}}_\lambda$ prohibits efficient training. To address this issue, we propose to accelerate mixup via *approxiate* GW geodesics. Nonetheless, it is hard to define approximate GW geodesics based on the original definition of GW geodesics. Hence, we will (i) reformulate the GW geodesic as a solution to an optimization problem and (ii) define approximate GW geodesics by introducing into the optimization problem a hyperparameter that controls the degree of approximation.

Theorem 3.2. *The tuple $(\boldsymbol{P}_1, \boldsymbol{P}_2, \tilde{\boldsymbol{\mu}}_{(\lambda)})$ in Eqs. (4) and (5) for exact GW geodesic is an optimal solution to the following optimization problem:*

$$\min_{\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{g}} \left(\varepsilon_{\boldsymbol{A}_1, \boldsymbol{A}_2}(\boldsymbol{P}_1 \mathrm{diag}(\boldsymbol{g}) \boldsymbol{P}_2^\top)\right)^{1/2} \quad \text{s.t.} \begin{cases} \boldsymbol{P}_1 \mathrm{diag}(\boldsymbol{g}) \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{g}) \\ \boldsymbol{P}_2 \mathrm{diag}(\boldsymbol{g}) \in \Pi(\boldsymbol{\mu}_2, \boldsymbol{g}) \\ \boldsymbol{g} \in \Delta_{n_1 n_2} \end{cases} \tag{7}$$

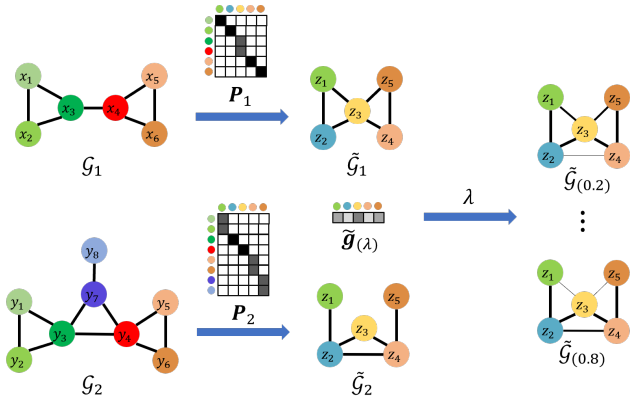


Figure 2. An example of GEOMIX transformation. Transformations $\boldsymbol{P}_1, \boldsymbol{P}_2$ map nodes $x_i \in \mathcal{G}_1, y_j \in \mathcal{G}_2$ into common latent nodes $z_k \in \tilde{\mathcal{G}}_1, \tilde{\mathcal{G}}_2$, and the well-aligned $\tilde{\mathcal{G}}_1, \tilde{\mathcal{G}}_2$ are linearly combined as the mixup graph $\tilde{\mathcal{G}}_{(\lambda)}$.

Proof is provided in Appendix A. Theorem 3.2 reformulates the GW geodesic as a solution to an optimization problem. Intuitively, by regarding the mixup graph as the latent factor, Eq. (7) factorizes the optimal coupling $\boldsymbol{T}$ into three parts, including: $\boldsymbol{P}_1, \boldsymbol{P}_2$ transforming $\mathcal{G}_1, \mathcal{G}_2$ to the latent graph $\mathcal{G}_{(\lambda)}$, and the node weight $\boldsymbol{g}$ for $\mathcal{G}_{(\lambda)}$.

However, the computation in Eq. (7) is still inefficient due to the high dimensionality $n_1 n_2$. To address this issue, we define *approximate* GW geodesics by replacing the dimensionality $n_1 n_2$ with a hyperparameter r that controls the degree of approximation.

To solve the optimization problem, we consider a change of variables: $\boldsymbol{Q}_1 = \boldsymbol{P}_1 \mathrm{diag}(\boldsymbol{g})$, $\boldsymbol{Q}_2 = \boldsymbol{P}_2 \mathrm{diag}(\boldsymbol{g})$, based on which the optimization problem is reformulated into a low-rank GW problem (Scetbon et al., 2022):

$$\operatorname*{arg\,min}_{\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g}} \left(\varepsilon_{\boldsymbol{A}_1, \boldsymbol{A}_2}(\boldsymbol{Q}_1 \mathrm{diag}(1/\boldsymbol{g}) \boldsymbol{Q}_2^\top)\right)^{1/2} \quad \text{s.t.} \begin{cases} \boldsymbol{Q}_1 \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{g}) \\ \boldsymbol{Q}_2 \in \Pi(\boldsymbol{\mu}_2, \boldsymbol{g}) \\ \boldsymbol{g} \in \Delta_r \end{cases} \tag{8}$$

In general, the optimal coupling $\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)$ is factorized into three parts $(\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g})$ that are closely connected to $(\boldsymbol{P}_1, \boldsymbol{P}_2, \tilde{\boldsymbol{\mu}}_{(\lambda)})$ for the exact GW geodesics in Eqs. (4) and (5). Specifically, $\boldsymbol{Q}_1, \boldsymbol{Q}_2$ approximates the unnormalized

Algorithm 1 GEOMIX

Input two graphs $\mathcal{G}_1 = \{\boldsymbol{A}_1, \boldsymbol{\mu}_1\}, \mathcal{G}_2 = \{\boldsymbol{A}_2, \boldsymbol{\mu}_2\}$, mixup graph size r, mixup ratio λ.
Initialize $\boldsymbol{g} = \frac{\mathbf{1}_r}{r}, \boldsymbol{Q}_1^{(0)} = \boldsymbol{\mu}_1 \otimes \boldsymbol{g}^\top, \boldsymbol{Q}_2^{(0)} = \boldsymbol{\mu}_2 \otimes \boldsymbol{g}^\top$;
for $t \in \mathbb{N}^+_{\leq T}$ **do**
  Compute $\boldsymbol{T}^{(t)}, \boldsymbol{\omega}^{(t)}, \boldsymbol{K}_1^{(t)}, \boldsymbol{K}_2^{(k)}, \boldsymbol{K}_3^{(t)}$ in Eq. (9);
  Solve $\underset{\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g}}{\arg\min}\, \mathrm{KL}\Big((\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g}), (\boldsymbol{K}_1^{(t)}, \boldsymbol{K}_2^{(t)}, \boldsymbol{K}_3^{(t)})\Big)$ by Dykstra's algorithm (Scetbon et al., 2021);
end for
Column-normalize $\boldsymbol{Q}_1^{(T)}, \boldsymbol{Q}_2^{(T)}$ as $\boldsymbol{P}_1, \boldsymbol{P}_2$;
Transform graphs $\tilde{\boldsymbol{A}}_1 = \boldsymbol{P}_1^\top \boldsymbol{A}_1 \boldsymbol{P}_1, \tilde{\boldsymbol{A}}_2 = \boldsymbol{P}_2^\top \boldsymbol{A}_2 \boldsymbol{P}_2$;
$\tilde{\mathcal{G}}_{(\lambda)} = \{(1-\lambda)\tilde{\boldsymbol{A}}_1 + \lambda \tilde{\boldsymbol{A}}_2, \mathrm{vec}(\boldsymbol{T}^{(T)})\}$;
$\tilde{\boldsymbol{y}}_{(\lambda)} = (1-\lambda)\boldsymbol{y}_1 + \lambda \boldsymbol{y}_2$;
return mixup graph $\tilde{\mathcal{G}}_{(\lambda)}$ and label $\tilde{\boldsymbol{y}}_{(\lambda)}$.

linear transformation $\boldsymbol{P}_1, \boldsymbol{P}_2$, and $\boldsymbol{g}$ approximates the node weight $\tilde{\boldsymbol{\mu}}_{(\lambda)}$. In our setting, we choose $r \leq n$ to map $\mathcal{G}_1, \mathcal{G}_2$ to the coarsen graphs $\tilde{\mathcal{G}}_1$ and $\tilde{\mathcal{G}}_2$ that are well-aligned. The intuition is that graphs often share common latent meanings at the coarsen granularity (Zhang et al., 2020). An illustrative example is shown in Figure 2.

To solve the optimization problem in Eq. (8), a mirror descent scheme w.r.t. the generalized KL divergence is proposed (Scetbon et al., 2022). In general, the mirror descent scheme iteratively solves the following problem:

$$\Big(\boldsymbol{Q}_1^{(t+1)}, \boldsymbol{Q}_2^{(t+1)}, \boldsymbol{g}^{(t+1)}\Big) = \underset{\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g}}{\arg\min}\, \mathrm{KL}\Big((\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g}), (\boldsymbol{K}_1^{(t)}, \boldsymbol{K}_2^{(t)}, \boldsymbol{K}_3^{(t)})\Big)$$
$$\text{s.t. } \boldsymbol{Q}_1 \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{g}),\ \boldsymbol{Q}_2 \in \Pi(\boldsymbol{\mu}_2, \boldsymbol{g}),\ \boldsymbol{g} \in \Delta_r$$
$$\text{where } \begin{cases} \boldsymbol{T}^{(t)} = \boldsymbol{Q}_1^{(t)} \mathrm{diag}(1/\boldsymbol{g}^{(k)}) \boldsymbol{Q}_2^{(t)^\top} \\ \boldsymbol{\omega}^{(t)}(i) = \Big(\boldsymbol{Q}_1^{(t)^\top} \boldsymbol{A}_1 \boldsymbol{T}^{(t)} \boldsymbol{A}_2 \boldsymbol{Q}_2^{(t)}\Big)(i,i) \\ \boldsymbol{K}_1^{(t)} = \exp\Big(4\gamma \boldsymbol{A}_1 \boldsymbol{T}^{(t)} \boldsymbol{A}_2 \boldsymbol{Q}_2^{(t)} \mathrm{diag}(1/\boldsymbol{g}^{(t)}) + \log \boldsymbol{Q}_1^{(t)}\Big) \\ \boldsymbol{K}_2^{(t)} = \exp\Big(4\gamma \boldsymbol{A}_2 \boldsymbol{T}^{(t)^\top} \boldsymbol{A}_1 \boldsymbol{Q}_1^{(t)} \mathrm{diag}(1/\boldsymbol{g}^{(t)}) + \log \boldsymbol{Q}_2^{(t)}\Big) \\ \boldsymbol{K}_3^{(t)} = \exp\Big(-4\gamma \boldsymbol{\omega}^{(t)}/\boldsymbol{g}^{(t)^2} + \log \boldsymbol{g}^{(t)}\Big) \end{cases} \tag{9}$$

where γ is the step size for mirror descent. In general, for each step t, we first find the unconstraint solution $\boldsymbol{K}_1^{(t)}$ to $\min_{\boldsymbol{Q}_1} \varepsilon(\boldsymbol{Q}_1 \mathrm{diag}(1/\boldsymbol{g}^{(t)} \boldsymbol{Q}_2^{(t)^\top})) + \mathrm{KL}(\boldsymbol{Q}_1, \boldsymbol{Q}_1^{(t)})$. Similarly, we can obtain the unconstraint solutions $\boldsymbol{K}_2^{(t)}, \boldsymbol{K}_3^{(t)}$ for $\boldsymbol{Q}_2, \boldsymbol{g}$, respectively. Then $\boldsymbol{K}_1^{(t)}, \boldsymbol{K}_2^{(t)}, \boldsymbol{K}_3^{(t)}$ are projected to the feasible regions $\Pi(\boldsymbol{\mu}_1, \boldsymbol{g}), \Pi(\boldsymbol{\mu}_2, \boldsymbol{g}), \Delta_r$, respectively, by minimizing the KL divergence. As proposed in (Scetbon et al., 2021), the above problem can be efficiently solve via the Dykstra's algorithm (Dykstra, 1983). Full algorithm details are provided in Appendix B. Afterwards, the transformations $\boldsymbol{P}_1, \boldsymbol{P}_2$ can be obtained by the column normalization of $\boldsymbol{Q}_1, \boldsymbol{Q}_2$. The overall algorithm is summarized in Algorithm 1.

On the factorization of the optimal coupling. The key idea of *transform-then-interpolate* strategy is to transform a graph pair into well-aligned pairs that are easy to interpolate. We show that the factorization $\boldsymbol{T} = \boldsymbol{P}_1 \mathrm{diag}(\boldsymbol{g}) \boldsymbol{P}_2^\top$ in Eq. (8) can be interpreted as a probabilistic alignment between $\boldsymbol{A}_1, \boldsymbol{A}_2$. Given that $\boldsymbol{P}_1, \boldsymbol{P}_2$ are column-normalized matrices, $\boldsymbol{P}_1(i,u) = p_1(i|u), \boldsymbol{P}_2(k,v) = p_2(k|v)$ indicate the generation probability of node $i \in \mathcal{G}_1, k \in \mathcal{G}_2$ conditioned on node $u, v \in \tilde{\mathcal{G}}_{(\lambda)}$, respectively. The simplex $\boldsymbol{g}(u) = p(u)$ indicates the prior probability of nodes $u \in \tilde{\mathcal{G}}_{(\lambda)}$. Then the factorization in Eq. (7) can be written as:

$$\begin{aligned} &\Big(\varepsilon_{\boldsymbol{A}_1, \boldsymbol{A}_2}(\boldsymbol{P}_1 \mathrm{diag}(\boldsymbol{g}) \boldsymbol{P}_2^\top)\Big)^{1/2} \\ &= \sum_{i,j,k,l} \Big((\boldsymbol{A}_1(i,j) - \boldsymbol{A}_2(k,l))^2 \boldsymbol{T}(i,k) \boldsymbol{T}(j,l)\Big)^{1/2} \\ &= \sum_{i,j,k,l} \Big((\boldsymbol{A}_1(i,j) - \boldsymbol{A}_2(k,l))^2 \boldsymbol{P}_1(i,u) \boldsymbol{g}(u) \boldsymbol{P}_2(k,u) \boldsymbol{P}_1(j,v) \boldsymbol{g}(v) \boldsymbol{P}_2(l,v)\Big)^{1/2} \\ &= \left(\mathbb{E}_{\substack{u \sim p(u),\ v \sim p(v) \\ i \sim p_1(\cdot|u),\ j \sim p_1(\cdot|v) \\ k \sim p_2(\cdot|u),\ l \sim p_2(\cdot|v)}} (\boldsymbol{A}_1(i,j) - \boldsymbol{A}_2(k,l))^2 \right)^{1/2} \end{aligned}$$

The above equation can be interpreted as a two-step alignment. We first select a node pair (u, v) from the mixup graph $\tilde{\mathcal{G}}_{(\lambda)}$ based on the prior probabilities $p(u)$ and $p(v)$. Then we select node pair i, j from $\mathcal{G}_1$ based on conditional probability $p_1(\cdot|u), p_1(\cdot|v)$, and node pair k, l from $\mathcal{G}_2$ based on conditional probability $p_2(\cdot|u), p_2(\cdot|v)$. Minimization of the above equation corresponds to finding the optimal transformations p_1, p_2, and node weight p to best align $\mathcal{G}_1, \mathcal{G}_2$.

Time complexity. Without loss of generality, we assume that input graphs share a comparable size that is greater than the mixup graph size r, i.e., $\mathcal{O}(n_1) = \mathcal{O}(n_2) = \mathcal{O}(n) > \mathcal{O}(r)$, we have the following time complexity analysis:

Proposition 3.3. *Given K graph pairs to be mixed up, the time complexity for* GEOMIX *is $\mathcal{O}(KTn^2r)$, where T is the number of iterations in the low-rank GW algorithm.*

The time complexity is quadratic w.r.t. the input graph size n and linear w.r.t. the mixup graph size r. It's worth mentioning that the exact GW geodesic requires $\mathcal{O}(n^4)$ time complexity, while the approximate GW geodesic, as we will empirically show in Section 4, achieves a significant reduction in running time with $\mathcal{O}(n^2r)$ time complexity at little expense of effectiveness.

4. Experiments

We conduct extensive experiments to evaluate the proposed GEOMIX. We first provide visualization on the mixup process and assess the geodesic property in Section 4.1. Then we evaluate the effectiveness of GEOMIX in enhancing GNN generalization and robustness in Sections 4.2 and 4.3, respectively. Further analysis is carried out in Section 4.4.

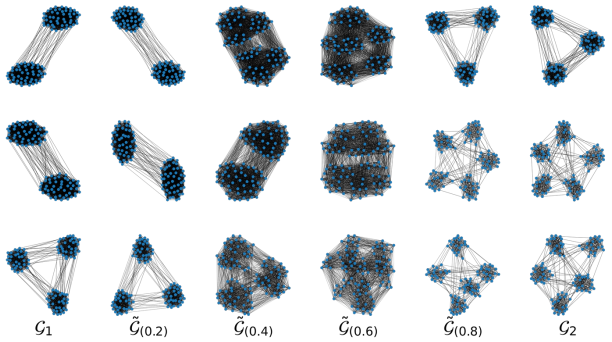
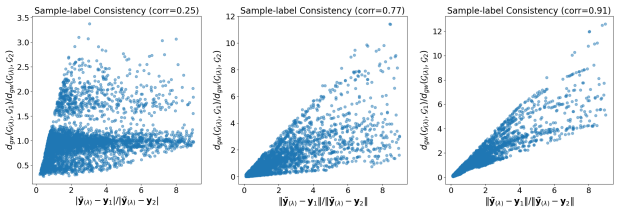
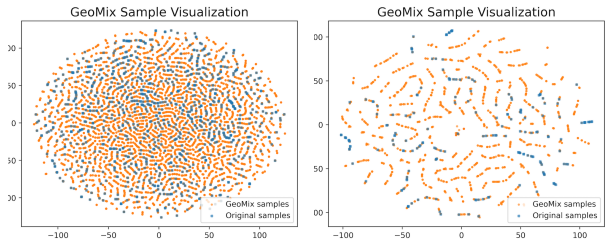


Figure 3. Visualization of the GEOMIX process. Each row corresponds to one mixup case. The leftmost and rightmost columns are input graphs, and the middle four columns are mixup graphs.



(a) G-Mixup (b) FGWMixup (c) GEOMIX

Figure 4. Sample–label consistency on IMDB-B. Each point represents a mixup sample–label pair. The Pearson coefficient between mixup samples and labels is 0.91 for GEOMIX, which is higher than those of G-Mixup (0.25) and FGWMixup (0.71).



(a) IMDB-B (b) MUTAG

Figure 5. Embedding visualization on (a) IMDB-B and (b) MUTAG. Mixup samples (orange) reside on the geodesics connecting original samples (blue).

4.1. Understanding the GEOMIX process

Visualize the mixup graphs. To understand the GEOMIX process, we first provide visualization of the mixup graphs in Figure 3. We use the Stochastic Block Model (SBM) to generate graphs with 2,3,5 blocks. GEOMIX is further employed to generate $\tilde{\mathcal{G}}_{(\lambda)}$ with $\lambda = \{0.2, 0.4, 0.6, 0.8\}$. It can be observed that GEOMIX indeed generates a mixture of two graphs, and higher the λ, more similar $\tilde{\mathcal{G}}_{(\lambda)}$ is to $\mathcal{G}_2$. More visualization results on real-world datasets are provided in Figure 9 in Appendix C.

Evaluate the sample–label consistency. We further evaluate whether the sample–label pairs generated by GEOMIX are consistent by evaluating the correlation between $\frac{d_{\mathrm{GW}}(\tilde{\mathcal{G}}_{(\lambda)},\mathcal{G}_1)}{d_{\mathrm{GW}}(\tilde{\mathcal{G}}_{(\lambda)},\mathcal{G}_2)}$ and $\frac{\|\tilde{\boldsymbol{y}}_{(\lambda)}-\boldsymbol{y}_1\|}{\|\tilde{\boldsymbol{y}}_{(\lambda)}-\boldsymbol{y}_2\|}$. We compare GEOMIX with two state-of-the-art graph mixup methods: G-Mixup (Han et al., 2022) and FGWMixup (Ma et al., 2023) on the IMDB-B dataset (Yanardag & Vishwanathan, 2015). As shown in Figure 4, $\frac{d_{\mathrm{GW}}(\tilde{\mathcal{G}}_{(\lambda)},\mathcal{G}_1)}{d_{\mathrm{GW}}(\tilde{\mathcal{G}}_{(\lambda)},\mathcal{G}_2)}$ is closely related to $\frac{\|\tilde{\boldsymbol{y}}_{(\lambda)}-\boldsymbol{y}_1\|}{\|\tilde{\boldsymbol{y}}_{(\lambda)}-\boldsymbol{y}_2\|}$ for GEOMIX achieving a Pearson coefficient of 0.91. For baseline methods like G-Mixup and FGWMixup, they achieve relatively lower Pearson coefficients of 0.25 and 0.71, respectively. This validates that samples generated by GEOMIX are consistent with their labels.

We also visualize the embeddings for IMDB-B and MUTAG datasets. Specifically, we use GCN model to learn graph embeddings for both original graphs (blue) and GEOMIX samples (orange), and adopt TSNE for visualization. As shown in Figure 5, the mixup samples form geodesics connecting the original graph pairs, which validates the geodesic property of GEOMIX. More embedding space visualization can be found in Figure 10 in Appendix C.

4.2. On the Generalization of GNNs

Experiment setup. To assess the effectiveness of GEOMIX in enhancing GNN generalization capability, we conduct experiments on graph classification on five real-world datasets, including PROTEINS (Borgwardt et al., 2005), MUTAG (Kriege & Mutzel, 2012), MSRC-9 (Neumann et al., 2016), IMDB-B (Yanardag & Vishwanathan, 2015), and IMDB-M (Yanardag & Vishwanathan, 2015). We split the dataset into train/test/validation set by 80%/10%/10% and use 10-fold cross validation for evaluation.

We consider two widely used GNN models, GCN (Kipf & Welling, 2017) and GIN (Xu et al., 2018), as the backbone models. We compare GEOMIX with eight state-of-the-art graph augmentation methods, including DropEdge (Rong et al., 2019), DropNode (You et al., 2020), Subgraph (You et al., 2020), M-Mixup (Wang et al., 2021), SubMix (Yoo et al., 2022), G-Mixup (Han et al., 2022), S-Mixup (Ling et al., 2023) and FGWMixup (Ma et al., 2023).

For a fair comparison, we adopt the same model architecture and hyperparameters. We use augmentation methods to generate the same number of augmented samples as the number of original samples. More experiment details can be found in Appendix D. Code and datasets are available at `https:`

Table 1. Experiment results on graph classification.

	MODEL	PROTEINS	MUTAG	MSRC-9	IMDB-B	IMDB-M
GCN	VANILLA	62.1±2.8	72.8±7.5	88.7±3.5	71.5±2.8	48.3±1.6
	DROPEDGE	61.6±2.9	73.3±7.0	89.4±4.2	71.4±2.7	47.8±1.9
	DROPNODE	61.1±2.6	71.3±9.0	89.0±4.0	71.6±3.4	47.7±1.8
	SUBGRAPH	59.6±2.6	70.2±9.5	83.1±3.2	71.7±3.7	45.5±2.8
	M-MIXUP	61.0±2.6	73.3±7.0	88.2±3.6	71.0±2.0	48.1±1.8
	SUBMIX	63.0±4.1	73.5±7.1	88.7±4.1	71.6±2.3	48.1±3.5
	G-MIXUP	63.6±3.5	73.3±7.0	12.2±5.3	**72.2±2.4**	48.0±2.5
	S-MIXUP	63.1±2.7	72.7±7.7	89.1±4.1	71.3±2.1	48.5±2.0
	FGWMIXUP	61.7±3.0	73.4±7.3	89.6±3.9	71.7±2.4	47.5±2.2
	GEOMIX	**70.2±4.7**	**76.1±5.5**	**90.3±2.9**	72.0±4.3	**49.3±5.5**
GIN	VANILLA	67.8±5.1	69.1±10.8	91.0±4.6	70.1±4.3	46.2±5.3
	DROPEDGE	66.8±4.9	71.2±7.4	88.2±5.8	70.0±4.0	46.5±4.0
	DROPNODE	66.8±4.6	71.8±6.7	91.8±4.9	71.6±3.6	44.9±5.0
	SUBGRAPH	62.3±4.9	75.0±7.6	85.6±10.3	68.1±4.8	45.4±4.9
	M-MIXUP	70.4±6.1	70.7±12.8	91.4±5.0	71.7±3.6	46.8±2.6
	SUBMIX	66.7±4.9	70.9±10.8	91.2±5.4	70.9±3.1	46.4±3.1
	G-MIXUP	69.6±6.5	73.4±9.7	19.5±9.1	71.1±3.4	**47.0±4.1**
	S-MIXUP	67.1±4.3	72.5±8.7	91.2±4.2	70.7±3.2	46.4±3.2
	FGWMIXUP	62.6±4.6	74.7±8.9	89.6±5.0	71.1±3.9	44.1±4.7
	GEOMIX	**71.3±3.7**	**80.8±6.2**	**92.8±3.6**	**72.1±2.9**	46.9±4.9

//github.com/zhichenz98/GeoMix-ICML24.

Results and analysis. The graph classification results are shown in Table 1. In general, our proposed GEOMIX achieves the best performance on 8 out 10 settings and the second-best performance on the rest 2 settings, with an up to 6.6% outperformance compared with the best competitor. Besides, we found that GEOMIX is more effective on datasets with fewer graphs and larger sizes (e.g., MUTAG and PROTEINS). This is because the mixup technique aims to generate synthetic samples covering the whole graph space. When dealing with larger and fewer graphs, the graph space is less covered, thus, the mixup technique achieves more significant improvements. This can be also validated by comparing the embedding visualization in Figure 5, where the MUTAG dataset is sparsely-covered while the IMDB-B dataset is more densely covered.

4.3. On the Robustness of GNNs

Experiment setup. Besides generalization, mixup can also improve the model robustness (Zhang et al., 2021). We evaluate model robustness against two kinds of corruption, namely topology corruption and label corruption. Specifically, for topology corruption, we randomly remove/add 10% or 20% of the edges; for label corruption, we randomly change the labels for 10% or 20% of the training graphs.

Results and analysis. Robustness results against topology and label corruption are shown in Tables 2 and 3, respectively. In general, our proposed GEOMIX improves model robustness against both topology and label corruption, achieving the best performance in most cases. Specifically, GEOMIX achieves up to 4.7% outperformance against topology corruption and 1.8% outperformance against label corruption compared with the best competitor. The reason for such outperformance is two-fold: (1) mixup samples that are far away from the training samples can help alleviate the model memorization of the corrupted labels (Zhang et al., 2018), and (2) the low-rank decomposition of GEOMIX naturally alleviates the topology noises, hence contributing to the robustness against topology noise.

Table 2. Robustness against topology corruption.

	MODEL	10%			20%		
		IMDB-B	IMDB-M	PROTEINS	IMDB-B	IMDB-M	PROTEINS
GCN	VANILLA	69.8±5.2	46.0±1.9	60.9±2.4	67.9±2.3	46.3±3.7	60.6±3.3
	DROPEDGE	70.3±3.0	44.9±2.8	60.8±3.6	69.8±4.0	46.7±3.2	60.2±2.3
	DROPNODE	69.1±3.6	46.7±3.2	61.2±2.7	69.8±3.2	47.3±3.2	59.3±2.8
	SUBGRAPH	69.3±2.9	44.1±5.0	59.6±2.6	68.4±3.4	46.3±4.3	59.6±2.6
	GEOMIX	**70.6±4.7**	**48.1±3.8**	**65.9±4.7**	**70.7±4.7**	**47.5±4.7**	**64.8±6.4**
GIN	VANILLA	66.4±4.4	**44.6±5.1**	63.2±5.5	67.6±3.7	41.9±3.9	59.6±2.6
	DROPEDGE	65.7±3.8	42.8±4.6	60.0±1.9	65.5±5.5	40.4±5.0	59.7±2.9
	DROPNODE	62.8±8.8	42.9±5.5	60.1±3.6	63.7±5.3	40.7±3.3	59.6±2.6
	SUBGRAPH	63.9±4.9	39.4±4.8	59.6±2.6	66.6±5.9	41.3±5.6	59.6±2.6
	GEOMIX	**67.1±4.6**	43.2±4.9	**66.3±6.3**	**67.8±3.6**	**42.4±5.3**	**61.2±6.2**

4.4. On the Effect of Mixup Graph Size

We analyze how the mixup graph size affect the effectiveness and efficiency of GEOMIX. As shown in Figure 7, the performance of GEOMIX is quite stable w.r.t. different mixup graph sizes, with at most $\pm 3\%$ fluctuation in accuracy. This indicates that (1) graphs contain redundant/noisy information and can be compressed as smaller graphs with lower r, and (2) the transformations employed by GEOMIX can capture the essential information for input graphs. Be-

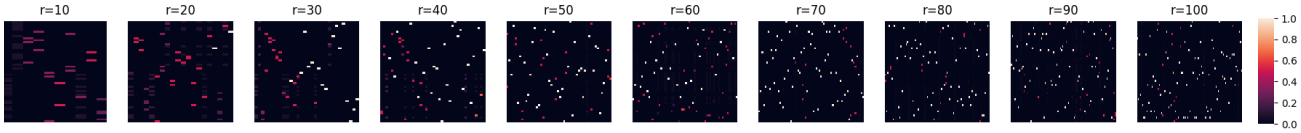


Figure 6. Transformation matrices with different mixup graph sizes r. Lighter the color, higher the value. As r increases, the transformation matrices become sparser and approximate the solutions to exact GW geodesic in Eq. (5).

Table 3. Robustness against label corruption.

	Model	10%			20%		
		IMDB-B	IMDB-M	PROTEINS	IMDB-B	IMDB-M	PROTEINS
GCN	Vanilla	57.5±8.8	40.6±2.7	53.9±4.4	55.0±9.8	36.0±8.0	50.0±7.8
	DropEdge	58.7±8.1	38.4±4.7	52.3±5.1	54.9±9.3	35.3±5.1	50.7±6.6
	DropNode	**60.6±8.3**	40.0±2.4	51.1±8.4	53.5±9.2	35.1±3.5	51.0±3.6
	Subgraph	57.4±9.3	36.7±5.8	49.8±5.9	55.0±5.8	35.1±4.7	50.6±4.1
	GeoMix	57.7±5.1	**41.2±5.0**	**55.3±6.6**	**55.7±9.1**	**37.8±3.2**	**51.3±3.2**
GIN	Vanilla	58.0±8.5	35.7±3.6	53.7±6.1	53.2±7.7	30.3±7.5	51.1±6.1
	DropEdge	54.7±8.8	37.8±5.8	54.6±9.2	52.1±3.9	37.2±5.6	**53.0±5.8**
	DropNode	**58.4±8.6**	35.7±4.9	53.4±4.6	52.8±6.4	34.7±6.3	51.9±5.6
	Subgraph	56.9±6.7	37.7±6.7	52.2±4.1	53.6±6.8	36.7±5.2	49.9±5.8
	GeoMix	58.0±7.5	**39.1±4.3**	**54.7±7.4**	**53.8±6.6**	**37.9±3.2**	52.2±6.6

sides, the running time of GeoMix grows approximately linearly w.r.t. r. Compared with the exact GW geodesic with $r = n^2$, a small mixup graph size r helps reduce the computational cost at little cost of effectiveness.

Besides, we visualize the learned transformations $\boldsymbol{P}_1$ with different mixup graph size r. Specifically, we consider two 2-block SBM graphs with graph size $n = 50$ and mixup size ranging from 10 to 100. Ideally, we expect the matrices to be column-wise sparse, i.e., each column to be a one-hot vector, so that it is analogous to the solution on exact GW geodesic in Eq. (5). As the results shown in Figure 6, the transformation matrices become sparser as r increases. This validates that (1) GeoMix provides good approximation to the exact GW geodesic solution in Eq. (5), and (2) the hyperparameter r achieves a tradeoff between the degree of approximation and algorithm efficiency.

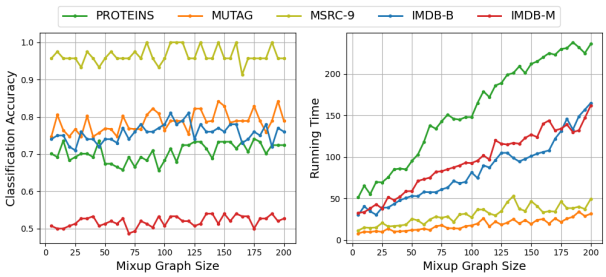


Figure 7. Hyperparameter study on mixup graph size r. Left: the classification accuracy is robust to different r. Right: the running time increase linearly w.r.t. r.

5. Related Work

Graph data augmentation. The superior performance of GNNs (Kipf & Welling, 2017; Xu et al., 2018; Gasteiger et al., 2018) on graph learning tasks largely depends on the quality of the training samples, which, however, are usually noisy and incomplete (Ding et al., 2022). Graph data augmentation has recently shown great power in enhancing model generalization and robustness, which can be categorized into three categories including, feature-based modification, structure-based modification and label-based modification. Feature-based methods manipulate node features by corruption (Feng et al., 2019; Yang et al., 2021), masking (You et al., 2020; 2021) and rewriting (Xu et al., 2022b), while structure-based methods perturb graph structure by node perturbation (You et al., 2020), edge perturbation (Rong et al., 2019; Xu et al., 2023a; Wei et al., 2022) and subgraph sampling (You et al., 2020). For label-based augmentation, a line of work interpolates graphs in the latent space (Wang et al., 2021; Verma et al., 2021; Navarro & Segarra, 2023), but may not be applicable to methods without latent space (e.g., kernel methods) and the mixup samples in the hidden space lack interpretability. Another line of work interpolates the aligned graph pairs (Guo & Mao, 2021; Ling et al., 2023; Kan et al., 2023), which can be regarded as a special case of our proposed GeoMix when fixing one transformation $\boldsymbol{P}$ as the identity matrix. Some other works adopt the transplant strategy by swapping the subgraphs of two graphs. More recently, FGWMixup (Ma et al., 2023) adopts the Fused Gromov–Wasserstein barycenter as the mixup graphs, but suffers from heavy computation.

Optimal transport on graphs. OT has achieved great success in processing geometric data, such as graphs, due to its ability in capturing the intrinsic geometry (Peyré et al., 2016) and comparing irregular objects in disparate spaces (Mémoli, 2011). For example, OT has been applied in different graph-learning tasks such as graph alignment (Xu et al., 2019; Zeng et al., 2023a; 2024), graph comparison (Maretic et al., 2019; Titouan et al., 2019), graph representation learning (Kolouri et al., 2021; Vincent-Cuaz et al., 2021; Zeng et al., 2023b), and many more. Besides, it is shown that the Gromov–Wasserstein space provides a joint space for disparate spaces, and the geodesic is well-defined on the GW space (Sturm, 2012), hence providing a natural solution to

geodesic-preserving graph mixup. In this work, we utilize the OT theory from two aspects. First, the GW distance is used as the fundamental metric to define the joint graph space as well as the geodesic for mixup (Sturm, 2012). Second, the OT coupling serves as the alignment between two graphs, whose decomposition transforms two graphs into a shared space for mixup (Scetbon et al., 2022).

Graph neural networks. Graph machine learning has shown great power in various real-world applications, such as fraud detection (Ding et al., 2021; Fu et al., 2022; Warmsley et al., 2022), social recommendation (Wei et al., 2020; Jing et al., 2022b; Wei & He, 2022; Jing et al., 2024), information retrieval (Fu & He, 2021; Yoo et al., 2023), and bioinformatics (Fu & He, 2022; Xu et al., 2024). Behind these applications, GNNs are the most prominent graph machine learning models with superior performance on graph classification (Xu et al., 2018), node classification (Yan et al., 2024c; Xu et al., 2022a; Liu et al., 2023b), node clustering (Jing et al., 2021a), link prediction (Yan et al., 2022; 2024b), and many more.

The key idea behind various GNNs is to leverage node information via custom aggregation and propagation. GCN (Kipf & Welling, 2017) aggregates neighboring node information via a localized first-order approximation of spectral graph convolutions. GraphSage (Hamilton et al., 2017) handles the inductive graph learning tasks by sampling and aggregating features from local neighborhood. GAT (Velickovic et al., 2017) adopts self-attention to provide more flexibility in the aggregation stage. GIN (Xu et al., 2018) achieves better expressiveness by leveraging the WL test. APPNP (Gasteiger et al., 2018) adopts personalized pagerank to achieve better node sampling. Recent studies attempt to generalize GNNs to more challenging settings, e.g., heterophilic graphs (Yan et al., 2024a; Xu et al., 2023b; Guo et al., 2023), heterogeneous graphs (Jing et al., 2021b; 2022a), dynamic graphs (Wang et al., 2023; Fu et al., 2020; 2023; Yan et al., 2021), and many more.

6. Conclusion

In this paper, we address the sample–label consistency issue in graph mixup from the geodesic perspective, and propose GeoMix to mixup graphs on the Gromov–Wasserstein geodesics. We first transform graph pairs into two well-aligned graphs through equivalence-preserving transformations, and then mixup on the exact GW geodesics. For faster computation, we parameterize the transformation by reformulating the GW geodesic as the solution to the low-rank GW problem, and an accelerated algorithm is further proposed. Extensive experiments demonstrate the effectiveness of the proposed GeoMix in enhancing GNN model generalization and robustness.

Impact Statement

This paper presents work whose goal is to advance the field of Graph Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

This work is supported by NSF (2134079, 2316233, 2324770, and 2117902), DARPA (HR001121C0165), NIFA (2020-67021-32799), DHS (17STQAC00001-07-00), AFOSR (FA9550-24-1-0002), the C3.ai Digital Transformation Institute, and IBM-Illinois Discovery Accelerator Institute. The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., and Kriegel, H.-P. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Chen, J., Yang, Z., and Yang, D. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. *arXiv preprint arXiv:2004.12239*, 2020.

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., and Hansch, C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34 (2):786–797, 1991.

Ding, K., Zhou, Q., Tong, H., and Liu, H. Few-shot network anomaly detection via cross-network meta-learning. In *Proceedings of the Web Conference 2021*, pp. 2448–2456, 2021.

Ding, K., Xu, Z., Tong, H., and Liu, H. Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter*, 24(2):61–77, 2022.

Dykstra, R. L. An algorithm for restricted least squares regression. *Journal of the American Statistical Association*, 78(384):837–842, 1983.

Feng, F., He, X., Tang, J., and Chua, T.-S. Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Transactions on Knowledge and Data Engineering*, 33(6):2493–2504, 2019.

Fu, D. and He, J. Sdg: A simplified and dynamic graph neural network. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2273–2277, 2021.

Fu, D. and He, J. Dppin: A biological repository of dynamic protein-protein interaction network data. In *2022 IEEE International Conference on Big Data (Big Data)*, pp. 5269–5277. IEEE, 2022.

Fu, D., Zhou, D., and He, J. Local motif clustering on time-evolving graphs. In *Proceedings of the 26th ACM SIGKDD International conference on knowledge discovery & data mining*, pp. 390–400, 2020.

Fu, D., Ban, Y., Tong, H., Maciejewski, R., and He, J. Disco: comprehensive and explainable disinformation detection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 4848–4852, 2022.

Fu, D., Xu, Z., Tong, H., and He, J. Natural and artificial dynamics in gnns: A tutorial. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pp. 1252–1255, 2023.

Gasteiger, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2018.

Guo, H. Nonlinear mixup: Out-of-manifold data augmentation for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4044–4051, 2020.

Guo, H. and Mao, Y. ifmixup: Towards intrusion-free graph mixup for graph classification. *arXiv e-prints*, pp. arXiv–2110, 2021.

Guo, H., Mao, Y., and Zhang, R. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*, 2019a.

Guo, H., Mao, Y., and Zhang, R. Mixup as locally linear out-of-manifold regularization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 3714–3722, 2019b.

Guo, K., Cao, X., Liu, Z., and Chang, Y. Taming over-smoothing representation on heterophilic graphs. *Information Sciences*, 647:119463, 2023.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Han, X., Jiang, Z., Liu, N., and Hu, X. G-mixup: Graph data augmentation for graph classification. In *International Conference on Machine Learning*, pp. 8230–8248. PMLR, 2022.

Howes, N. R. *Modern Analysis and Topology*. Springer Science & Business Media, 1995.

Jing, B., Park, C., and Tong, H. Hdmi: High-order deep multiplex infomax. In *Proceedings of the Web Conference 2021*, pp. 2414–2424, 2021a.

Jing, B., Tong, H., and Zhu, Y. Network of tensor time series. In *Proceedings of the Web Conference 2021*, pp. 2425–2437, 2021b.

Jing, B., Feng, S., Xiang, Y., Chen, X., Chen, Y., and Tong, H. X-goal: Multiplex heterogeneous graph prototypical contrastive learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 894–904, 2022a.

Jing, B., Yan, Y., Zhu, Y., and Tong, H. Coin: Co-cluster infomax for bipartite graphs. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022b.

Jing, B., Yan, Y., Ding, K., Park, C., Zhu, Y., Liu, H., and Tong, H. Sterling: Synergistic representation learning on bipartite graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12976–12984, 2024.

Kan, X., Li, Z., Cui, H., Yu, Y., Xu, R., Yu, S., Zhang, Z., Guo, Y., and Yang, C. R-mixup: Riemannian mixup for biological networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 1073–1085, 2023.

Kim, J.-H., Choo, W., and Song, H. O. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning*, pp. 5275–5285. PMLR, 2020.

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

Kolouri, S., Naderializadeh, N., Rohde, G. K., and Hoffmann, H. Wasserstein embedding for graph learning. In *International Conference on Learning Representations*, 2021.

Kriege, N. and Mutzel, P. Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483*, 2012.

Ling, H., Jiang, Z., Liu, M., Ji, S., and Zou, N. Graph mixup with soft alignments. In *International Conference on Machine Learning*, volume 202, pp. 21335–21349. PMLR, 2023.

Liu, Z., Li, S., Wang, G., Wu, L., Tan, C., and Li, S. Z. Harnessing hard mixed samples with decoupled regularizer. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a.

Liu, Z., Zeng, Z., Qiu, R., Yoo, H., Zhou, D., Xu, Z., Zhu, Y., Weldemariam, K., He, J., and Tong, H. Topological augmentation for class-imbalanced node classification. *arXiv preprint arXiv:2308.14181*, 2023b.

Ma, X., Chu, X., Wang, Y., Lin, Y., Zhao, J., Ma, L., and Zhu, W. Fused gromov-wasserstein graph mixup for graph-level classifications. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Maretic, H. P., El Gheche, M., Chierchia, G., and Frossard, P. Got: an optimal transport framework for graph comparison. *Advances in Neural Information Processing Systems*, 32, 2019.

Mémoli, F. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of Computational Mathematics*, 11:417–487, 2011.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

Navarro, M. and Segarra, S. Graphmad: Graph mixup for data augmentation using data-driven convex clustering. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

Neumann, M., Garnett, R., Bauckhage, C., and Kersting, K. Propagation kernels: efficient graph kernels from propagated information. *Machine learning*, 102:209–245, 2016.

Peyré, G., Cuturi, M., and Solomon, J. Gromov-wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*, pp. 2664–2672. PMLR, 2016.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

Scetbon, M., Cuturi, M., and Peyré, G. Low-rank sinkhorn factorization. In *International Conference on Machine Learning*, pp. 9344–9354. PMLR, 2021.

Scetbon, M., Peyré, G., and Cuturi, M. Linear-time gromov wasserstein distances using low rank couplings and costs. In *International Conference on Machine Learning*, pp. 19347–19365. PMLR, 2022.

Sturm, K.-T. The space of spaces: curvature bounds and gradient flows on the space of metric measure spaces. *arXiv preprint arXiv:1208.0434*, 2012.

Titouan, V., Courty, N., Tavenard, R., and Flamary, R. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*, pp. 6275–6284. PMLR, 2019.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., et al. Graph attention networks. *stat*, 1050 (20):10–48550, 2017.

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., and Bengio, Y. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*, pp. 6438–6447. PMLR, 2019.

Verma, V., Qu, M., Kawaguchi, K., Lamb, A., Bengio, Y., Kannala, J., and Tang, J. Graphmix: Improved training of gnns for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10024–10032, 2021.

Vincent-Cuaz, C., Vayer, T., Flamary, R., Corneli, M., and Courty, N. Online graph dictionary learning. In *International conference on machine learning*, pp. 10564–10574. PMLR, 2021.

Wang, D., Yan, Y., Qiu, R., Zhu, Y., Guan, K., Margenot, A., and Tong, H. Networked time series imputation via position-aware graph enhanced variational autoencoders. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2256–2268, 2023.

Wang, Y., Wang, W., Liang, Y., Cai, Y., and Hooi, B. Mixup for node and graph classification. In *Proceedings of the Web Conference 2021*, pp. 3663–3674, 2021.

Warmsley, D., Waagen, A., Xu, J., Liu, Z., and Tong, H. A survey of explainable graph neural networks for cyber malware analysis. In *2022 IEEE International Conference on Big Data (Big Data)*, pp. 2932–2939. IEEE, 2022.

Wei, T. and He, J. Comprehensive fair meta-learned recommender system. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1989–1999, 2022.

Wei, T., Wu, Z., Li, R., Hu, Z., Feng, F., He, X., Sun, Y., and Wang, W. Fast adaptation for cold-start collaborative filtering with meta-learning. In *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 661–670. IEEE, 2020.

Wei, T., You, Y., Chen, T., Shen, Y., He, J., and Wang, Z. Augmentations in hypergraph contrastive learning: Fabricated and generative. *Advances in neural information processing systems*, 35:1909–1922, 2022.

Wei, T., Guo, Z., Chen, Y., and He, J. Ntk-approximating mlp fusion for efficient language model fine-tuning. In *International Conference on Machine Learning*, pp. 36821–36838. PMLR, 2023.

Wei, T., Jin, B., Li, R., Zeng, H., Wang, Z., Sun, J., Yin, Q., Lu, H., Wang, S., He, J., et al. Towards unified multimodal personalization: Large vision-language models for generative recommendation and beyond. *arXiv preprint arXiv:2403.10667*, 2024.

Xu, H., Luo, D., and Carin, L. Scalable gromov-wasserstein learning for graph partitioning and matching. *Advances in Neural Information Processing Systems*, 32, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

Xu, Z., Ding, K., Wang, Y.-X., Liu, H., and Tong, H. Generalized few-shot node classification. In *2022 IEEE International Conference on Data Mining (ICDM)*, pp. 608–617. IEEE, 2022a.

Xu, Z., Du, B., and Tong, H. Graph sanitation with application to node classification. In *Proceedings of the ACM Web Conference 2022*, pp. 1136–1147, 2022b.

Xu, Z., Chen, Y., Pan, M., Chen, H., Das, M., Yang, H., and Tong, H. Kernel ridge regression-based graph dataset distillation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2850–2861, 2023a.

Xu, Z., Chen, Y., Zhou, Q., Wu, Y., Pan, M., Yang, H., and Tong, H. Node classification beyond homophily: Towards a general solution. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2862–2873, 2023b.

Xu, Z., Qiu, R., Chen, Y., Chen, H., Fan, X., Pan, M., Zeng, Z., Das, M., and Tong, H. Discrete-state continuous-time diffusion for graph generation. *arXiv preprint arXiv:2405.11416*, 2024.

Yan, Y., Liu, L., Ban, Y., Jing, B., and Tong, H. Dynamic knowledge graph alignment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 4564–4572, 2021.

Yan, Y., Zhou, Q., Li, J., Abdelzaher, T., and Tong, H. Dissecting cross-layer dependency inference on multi-layered inter-dependent networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 2341–2351, 2022.

Yan, Y., Chen, Y., Chen, H., Xu, M., Das, M., Yang, H., and Tong, H. From trainable negative depth to edge heterophily in graphs. *Advances in Neural Information Processing Systems*, 36, 2024a.

Yan, Y., Hu, Y., Zhou, Q., Liu, L., Zeng, Z., Chen, Y., Pan, M., Chen, H., Das, M., and Tong, H. Pacer: Network embedding from positional to structural. In *Proceedings of the ACM on Web Conference 2024*, pp. 2485–2496, 2024b.

Yan, Y., Jing, B., Liu, L., Wang, R., Li, J., Abdelzaher, T., and Tong, H. Reconciling competing sampling strategies of network embedding. *Advances in Neural Information Processing Systems*, 36, 2024c.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, 2015.

Yang, L., Zhang, L., and Yang, W. Graph adversarial self-supervised learning. *Advances in Neural Information Processing Systems*, 34:14887–14899, 2021.

Yoo, H., Zeng, Z., Kang, J., Liu, Z., Zhou, D., Wang, F., Chan, E., and Tong, H. Ensuring user-side fairness in dynamic recommender systems. *arXiv preprint arXiv:2308.15651*, 2023.

Yoo, J., Shim, S., and Kang, U. Model-agnostic augmentation for accurate graph classification. In *Proceedings of the ACM Web Conference 2022*, pp. 1281–1291, 2022.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020.

You, Y., Chen, T., Shen, Y., and Wang, Z. Graph contrastive learning automated. In *International Conference on Machine Learning*, pp. 12121–12132. PMLR, 2021.

Zeng, Z., Zhang, S., Xia, Y., and Tong, H. Parrot: Position-aware regularized optimal transport for network alignment. In *Proceedings of the ACM Web Conference 2023*, pp. 372–382, 2023a.

Zeng, Z., Zhu, R., Xia, Y., Zeng, H., and Tong, H. Generative graph dictionary learning. In *International Conference on Machine Learning*, pp. 40749–40769. PMLR, 2023b.

Zeng, Z., Du, B., Zhang, S., Xia, Y., Liu, Z., and Tong, H. Hierarchical multi-marginal optimal transport for network alignment. In *Proceedings of the AAAI Conference*

on Artificial Intelligence, volume 38, pp. 16660–16668, 2024.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Zhang, L., Deng, Z., Kawaguchi, K., Ghorbani, A., and Zou, J. How does mixup help with robustness and generalization? In *International Conference on Learning Representations*, 2021.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

Zhang, S., Tong, H., Xia, Y., Xiong, L., and Xu, J. Nettrans: Neural cross-network transformation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 986–996, 2020.

Appendix

The content of the appendix is organized as follows:

- Appendix A includes the proofs for all the theorems in the main content.
- Appendix B provides the details of GEOMIX and discuss potential variants:
  - Appendix B.1 provides details for the Dykstra's and Low-rank GW algorithm.
  - Appendix B.2 generalizes GEOMIX to attributed graphs with node attributes.
- Appendix C provides additional experimental results, including sensitivity analysis, mixup graph visualization, and embedding space visualization.
- Appendix D details the reproducibility, including dataset description, model architecture, and experiment pipeline.
- Appendix E discusses the limitations and possible future directions of this work.

A. Proof

Theorem 3.1. *Given two graphs $\mathcal{G}_1, \mathcal{G}_2$, the transformed graphs $\tilde{\mathcal{G}}_1, \tilde{\mathcal{G}}_2$ in Eq. (5) are in the equivalent class of $\mathcal{G}_1, \mathcal{G}_2$, respectively, that is*

$$[\![\tilde{\mathcal{G}}_1]\!] = [\![\mathcal{G}_1]\!], [\![\tilde{\mathcal{G}}_2]\!] = [\![\mathcal{G}_2]\!]$$

Furthermore, the curve $\{[\![\tilde{\mathcal{G}}_{(\lambda)}]\!]\}_{0\leq\lambda\leq 1}$ is a GW geodesic connecting $[\![\mathcal{G}_1]\!]$ and $[\![\mathcal{G}_2]\!]$, which means that for every $\lambda_1, \lambda_2 \in [0,1]$,

$$d^*_{\text{GW}}([\![\tilde{\mathcal{G}}_{(\lambda_1)}]\!], [\![\tilde{\mathcal{G}}_{(\lambda_2)}]\!]) = |\lambda_1 - \lambda_2| \cdot d^*_{\text{GW}}([\![\mathcal{G}_1]\!], [\![\mathcal{G}_2]\!]).$$

Proof. We first prove that $\mathcal{G}_1 \sim \tilde{\mathcal{G}}_1$ in terms of the GW distance. According to Eq. (2), the GW distance between $\mathcal{G}_1$ and $\tilde{\mathcal{G}}_1$ can be written as follows:

$$d_{\text{GW}}(\mathcal{G}_1, \tilde{\mathcal{G}}_1) = \underset{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_1,\tilde{\boldsymbol{\mu}}_1)}{\arg\min} \sum_{\substack{i,j \\ (x_1,y_1),(x_2,y_2)}} \left[\boldsymbol{A}_1(i,j) - \tilde{\boldsymbol{A}}_1((x_1,y_1),(x_2,y_2))\right]^2 \boldsymbol{T}(i,(x_1,y_1))\boldsymbol{T}(j,(x_2,y_2))$$

It is easy to verify that the marginal distributions satisfy $\tilde{\boldsymbol{\mu}}_1 = \tilde{\boldsymbol{P}}_1\boldsymbol{\mu}_1 = \boldsymbol{\mu}_1 \otimes \frac{\mathbf{1}_{n_2}}{n_2}$ and $\tilde{\boldsymbol{\mu}}_2 = \tilde{\boldsymbol{P}}_2\boldsymbol{\mu}_2 = \boldsymbol{\mu}_2 \otimes \frac{\mathbf{1}_{n_1}}{n_1}$. We consider a naive coupling $\boldsymbol{T}_1 \in \Pi(\boldsymbol{\mu}_1, \tilde{\boldsymbol{\mu}}_1)$ as follows:

$$\boldsymbol{T}_1(i,(x_1,y_1)) = \begin{cases} \dfrac{\boldsymbol{\mu}_1(i)}{n_2}, & \text{if } i = x_1 \\ 0, & \text{else} \end{cases}, \tag{10}$$

which leads to a (sub)optimal solution to the GW distance, that is:

$$\begin{aligned}
d_{\text{GW}}(\mathcal{G}_1, \tilde{\mathcal{G}}_1) \leq & \sum_{\substack{i,j \\ (x_1,y_1),(x_2,y_2)}} \left[\boldsymbol{A}_1(i,j) - \tilde{\boldsymbol{A}}_1((x_1,y_1),(x_2,y_2))\right]^2 \boldsymbol{T}_1(i,(x_1,y_1))\boldsymbol{T}_1(j,(x_2,y_2)) \\
= & \sum_{\substack{i,j \\ (x_1,y_1),(x_2,y_2)}} \left[\boldsymbol{A}_1(i,j) - \sum_{i,j} \boldsymbol{P}_1(i,(x_1,y_1))\boldsymbol{A}_1(i,j)\boldsymbol{P}_1(j,(x_2,y_2))\right]^2 \boldsymbol{T}_1(i,(x_1,y_1))\boldsymbol{T}_1(j,(x_2,y_2)) \\
= & \sum_{\substack{i,j \\ (x_1,y_1),(x_2,y_2)}} \left[\boldsymbol{A}_1(i,j) - \boldsymbol{A}_1(x_1,x_2)\right]^2 \boldsymbol{T}_1(i,(x_1,y_1))\boldsymbol{T}_1(j,(x_2,y_2)) \quad \text{due to Eq. (5)} \\
= & \sum_{i,j} \left[\boldsymbol{A}_1(i,j) - \boldsymbol{A}_1(i,j)\right]^2 \frac{\boldsymbol{\mu}_1(i)\boldsymbol{\mu}_1(j)}{n_2^2} \quad \text{due to Eq. (10)} \\
= & \, 0
\end{aligned}$$

Since the GW distance is non-negative, we have $d_{\mathrm{GW}}(\mathcal{G}_1, \tilde{\mathcal{G}}_1) = 0$. Similarly, we can also show that $d_{\mathrm{GW}}(\mathcal{G}_2, \tilde{\mathcal{G}}_2) = 0$. Therefore, we prove $\mathcal{G}_1 \sim \tilde{\mathcal{G}}_1, \mathcal{G}_2 \sim \tilde{\mathcal{G}}_2$. Then we show that the mixup graph $\mathcal{G}_{(\lambda)} = \big(\boldsymbol{A}_{(\lambda)}, \boldsymbol{\mu}_{(\lambda)}\big)$ defines a geodesic connecting $\mathcal{G}_1$ and $\mathcal{G}_2$ in the GW space. We can write down $\boldsymbol{A}_{(\lambda)}((x_1, y_1), (x_2, y_2))$ as:

$$\begin{aligned}\boldsymbol{A}_\lambda((x_1, y_1), (x_2, y_2)) &= (1-\lambda) \sum_{i,j \in \mathcal{G}_1} \underbrace{\boldsymbol{P}_1(i, (x_1, y_1))}_{1[i=x_1]} \boldsymbol{A}_1(i,j) \underbrace{\boldsymbol{P}_1(j, (x_2, y_2))}_{1[j=x_2]} + \lambda \sum_{i,j \in \mathcal{G}_2} \underbrace{\boldsymbol{P}_2(i, (x_1, y_1))}_{1[i=y_1]} \boldsymbol{A}_2(i,j) \underbrace{\boldsymbol{P}_2(j, (x_2, y_2))}_{1[j=y_2]} \\ &= (1-\lambda)\boldsymbol{A}_1(x_1, x_2) + \lambda \boldsymbol{A}_2(y_1, y_2)\end{aligned}$$

Since $\boldsymbol{\mu}_{(\lambda)} = (1-\lambda)\tilde{\boldsymbol{\mu}}_1 + \lambda \tilde{\boldsymbol{\mu}}_2 = \mathrm{vecOT}(\mathcal{G}_1, \mathcal{G}_2)$ in Eqs. (5) and (6), according to Eq. (4), $\mathcal{G}_{(\lambda)}$ corresponds to the geodesic connecting $\mathcal{G}_1, \mathcal{G}_2$ in the GW space. □

Theorem 3.2. *The tuple $(\boldsymbol{P}_1, \boldsymbol{P}_2, \tilde{\boldsymbol{\mu}}_{(\lambda)})$ in Eqs. (4) and (5) for exact GW geodesic is an optimal solution to the following optimization problem:*

$$\begin{aligned}&\min_{\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{g}} \left(\varepsilon_{\boldsymbol{A}_1, \boldsymbol{A}_2}(\boldsymbol{P}_1 \mathrm{diag}(\boldsymbol{g}) \boldsymbol{P}_2^\intercal)\right)^{1/2} \\ &\text{s.t.} \begin{cases} \boldsymbol{P}_1 \mathrm{diag}(\boldsymbol{g}) \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{g}) \\ \boldsymbol{P}_2 \mathrm{diag}(\boldsymbol{g}) \in \Pi(\boldsymbol{\mu}_2, \boldsymbol{g}) \\ \boldsymbol{g} \in \Delta_{n_1 n_2} \end{cases}\end{aligned} \tag{7}$$

Proof. We first show that $(\boldsymbol{P}_1, \boldsymbol{P}_2, \tilde{\boldsymbol{\mu}}_{(\lambda)})$ satisfies the constraints of the optimization problem. By definition in Eq. (4), we have $\boldsymbol{g} = \tilde{\boldsymbol{\mu}}_{(\lambda)} = \mathrm{vec}(\boldsymbol{T}) \in \Delta_{n_1 n_2}$, where $\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)$ is the optimal coupling between $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ in terms of the GW distance. Note that

$$\begin{aligned}\sum_i (\boldsymbol{P}_1 \mathrm{diag}(\boldsymbol{g}))(x_i, (x_j, y_k)) &= \sum_i \boldsymbol{P}_1(x_i, (x_j, y_k)) \boldsymbol{g}((x_j, y_k)) = \boldsymbol{T}(x_j, y_k) \\ \sum_{j,k} (\boldsymbol{P}_1 \mathrm{diag}(\boldsymbol{g}))(x_i, (x_j, y_k)) &= \sum_{j,k} \boldsymbol{P}_1(x_i, (x_j, y_k)) \boldsymbol{g}((x_j, y_k)) = \sum_k \boldsymbol{T}(x_i, y_k) = \boldsymbol{\mu}_1(x_i)\end{aligned}$$

Therefore, we have $\boldsymbol{P}_1 \mathrm{diag}(\boldsymbol{g}) \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{g})$. Similarly, we can show $\boldsymbol{P}_2 \mathrm{diag}(\boldsymbol{g}) \in \Pi(\boldsymbol{\mu}_2, \boldsymbol{g})$.

To validate that $\boldsymbol{P}_1 \mathrm{diag}(\tilde{\boldsymbol{\mu}}_{(\boldsymbol{\lambda})}) \boldsymbol{P}_2^\intercal$ is an optimal coupling between $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$, it suffices to show that $\boldsymbol{P}_1 \mathrm{diag}(\tilde{\boldsymbol{\mu}}_{(\boldsymbol{\lambda})}) \boldsymbol{P}_2^\intercal = \mathrm{vec}(\boldsymbol{T})$ as follows

$$\begin{aligned}(\boldsymbol{P}_1 \mathrm{diag}(\mathrm{vec}(\boldsymbol{T})) \boldsymbol{P}_2^\intercal)(x_i, y_k) &= \sum_{j,l} \boldsymbol{P}_1(x_i, (x_j, y_l)) \mathrm{vec}(\boldsymbol{T})((x_j, y_l)) \boldsymbol{P}_2(y_k, (x_j, y_l)) \\ &= \sum_{j,l} 1[i=j] \boldsymbol{T}(x_j, y_l) 1[k=l] \\ &= \boldsymbol{T}(x_i, y_k)\end{aligned}$$

Therefore, we prove that $\boldsymbol{P}_1 \mathrm{diag}(\mathrm{vec}(\boldsymbol{T})) \boldsymbol{P}_2^\intercal$ is exactly the optimal coupling between $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. In other words, the tuple $(\boldsymbol{P}_1, \boldsymbol{P}_2, \tilde{\boldsymbol{\mu}}_{(\lambda)})$ for exact GW geodesic corresponds to the solution to the given optimization problem. □

B. Algorithm

B.1. Detailed Algorithm

In this section, we present the Dykstra's algorithm (Scetbon et al., 2021) and low-rank Gromov–Wasserstein (Scetbon et al., 2022) for completeness.

Dykstra's algorithm (Dykstra, 1983; Scetbon et al., 2021). Given $\boldsymbol{\xi}_1 \in \mathbb{R}_+^{n_1 \times r}, \boldsymbol{\xi}_2 \in \mathbb{R}_+^{n_2 \times r}, \boldsymbol{\xi}_3 \in \mathbb{R}_+^r$, and a convex set $\mathcal{C} \subset \mathbb{R}_+^{n_1 \times r} \times \mathbb{R}_+^{n_2 \times r} \times \mathbb{R}_+^r$, the Dykstra's algorithm seeks for a set of the optimal variables $(\boldsymbol{Q}_1^*, \boldsymbol{Q}_2^*, \boldsymbol{g}^*) \in \mathcal{C}$ that is closest to $(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3)$ in terms of the generalized KL divergence, that is:

$$(\boldsymbol{Q}_1^*, \boldsymbol{Q}_2^*, \boldsymbol{g}^*) = \underset{(\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g}) \subset \mathcal{C}}{\arg\min} \ \mathrm{KL}((\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g}), (\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3))$$

Algorithm 2 Dykstra's Algorithm (LR-Dykstra) (Scetbon et al., 2021)

Input target values $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3$, marginals $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$, lower bound α, error threshold δ
Initialize scaling vectors $\boldsymbol{q}_1 = \boldsymbol{q}_2 = \boldsymbol{q}_3^{(1)} = \boldsymbol{q}_3^{(2)} = \tilde{\boldsymbol{v}}_1 = \tilde{\boldsymbol{v}}_2 = \mathbf{1}_r, \tilde{\boldsymbol{g}} = \boldsymbol{\xi}_3$;
while $\sum_{i=1}^{2} \|\boldsymbol{u}_i \odot \boldsymbol{\xi}_i \boldsymbol{v}_i - \boldsymbol{\mu}_i\|_1 \geq \delta$ **do**
  Update row-normalization vectors $\boldsymbol{u}_i = \boldsymbol{\mu}_i / \boldsymbol{\xi}_i \tilde{\boldsymbol{v}}_i, \forall i = \{1, 2\}$;
  Stabilize $\boldsymbol{g} = \max(\alpha, \tilde{\boldsymbol{g}} \odot \boldsymbol{q}_3^{(1)})$, update scale vector $\boldsymbol{q}_3^{(1)} = (\tilde{\boldsymbol{g}} \odot \boldsymbol{q}_3^{(1)})/\boldsymbol{g}$, and update $\tilde{\boldsymbol{g}} = \boldsymbol{g}$;
  Update $\boldsymbol{g} = (\tilde{\boldsymbol{g}} \odot \boldsymbol{q}_3^{(2)})^{1/3} \prod_{i=1}^{2} (\boldsymbol{v}_i \odot \boldsymbol{q}_i \odot \boldsymbol{\xi}_i^\intercal \boldsymbol{u}_i)^{1/3}$;
  Update column-normalization vectors $\boldsymbol{v}_i = \boldsymbol{g} / \boldsymbol{\xi}_i^\intercal \boldsymbol{u}_i, \forall i \in \{1, 2\}$;
  Update scale vectors $\boldsymbol{q}_i = (\tilde{\boldsymbol{v}}_i \odot \boldsymbol{q}_i)/\boldsymbol{v}_i, \tilde{\boldsymbol{v}}_i = \boldsymbol{v}_i, \forall i \in \{1, 2\}, \boldsymbol{q}_3^{(2)} = (\tilde{\boldsymbol{g}} \odot \boldsymbol{q}_3^{(2)})/\boldsymbol{g}, \tilde{\boldsymbol{g}} = \boldsymbol{g}$;
end while
Compute $\boldsymbol{Q}_1 = \text{diag}(\boldsymbol{u}_1)\boldsymbol{\xi}_1 \text{diag}(\boldsymbol{v}_1), \boldsymbol{Q}_2 = \text{diag}(\boldsymbol{u}_2)\boldsymbol{\xi}_2 \text{diag}(\boldsymbol{v}_2)$;
return $(\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g})$;

Algorithm 3 Low-rank Gromov–Wasserstein (Scetbon et al., 2022)

Input cost matrices $\boldsymbol{A}_1, \boldsymbol{A}_2$, marginals $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$, rank r, step size γ, lower bound α, error threshold δ.
Initialize $\boldsymbol{x}_1 = \boldsymbol{A}_1^{\odot 2} \boldsymbol{\mu}_1, \boldsymbol{x}_2 = \boldsymbol{A}_2^{\odot 2} \boldsymbol{\mu}_2, \boldsymbol{z}_1 = \boldsymbol{x}_1^{\odot 2}, \boldsymbol{z}_2 = \boldsymbol{x}_2^{\odot 2}, \boldsymbol{g} \in \Delta_r, \boldsymbol{Q}_1 \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{g}), \boldsymbol{Q}_2 \Pi(\boldsymbol{\mu}_2, \boldsymbol{g})$;
for $t = 1, 2, \ldots$ **do**
  $\boldsymbol{\xi}_1 = \boldsymbol{Q}_1 \odot \exp(-4\gamma \boldsymbol{A}_1 \boldsymbol{Q}_1 \text{diag}(1/\boldsymbol{g}) \boldsymbol{Q}_2^\intercal \boldsymbol{A}_2 \boldsymbol{Q}_2 \text{diag}(1/\boldsymbol{g}))$;
  $\boldsymbol{\xi}_2 = \boldsymbol{Q}_2 \odot \exp(-4\gamma \boldsymbol{A}_2^\intercal \boldsymbol{Q}_2 \text{diag}(1/\boldsymbol{g}) \boldsymbol{Q}_1^\intercal \boldsymbol{A}_1^\intercal \boldsymbol{Q}_1 \text{diag}(1/\boldsymbol{g}))$;
  $\boldsymbol{\omega} = \text{diag}(-\boldsymbol{Q}_1^\intercal \boldsymbol{A}_1 \boldsymbol{Q}_1 \text{diag}(1/\boldsymbol{g}) \boldsymbol{Q}_2^\intercal \boldsymbol{A}_2 \boldsymbol{Q}_2)$;
  $\boldsymbol{\xi}_3 = \boldsymbol{g} \odot \exp(-4\gamma \boldsymbol{\omega}/\boldsymbol{g}^2)$;
  $\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g} = \text{LR-Dykstra}(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \alpha, \delta)$;
end for
return $(\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g})$.

The general idea of Dykstra's algorithm is to rescale $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2$ and project onto the convex hull $\mathcal{C}$. By iterating the above process, Dykstra's algorithm is shown to converge to the desired solution (Dykstra, 1983). The Dykstra's algorithm is shown in Algorithm 2.

Low-rank Gromov–Wasserstein (Scetbon et al., 2022). Given two adjacency matrices $\boldsymbol{A}_1 \in \mathbb{R}^{n_1 \times n_1}, \boldsymbol{A}_2 \in \mathbb{R}^{n_2 \times n_2}$, the low-rank GW problem seeks for the optimal low-rank decomposition of the optimal coupling $\boldsymbol{T} = \boldsymbol{Q}_1 \text{diag}(1/\boldsymbol{g}) \boldsymbol{Q}_2^\intercal$ to the GW distance, that is](Scetbon et al., 2022),

$$(\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g}) = \underset{\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g}}{\arg\min} (\varepsilon_{\boldsymbol{A}_1, \boldsymbol{A}_2}(\boldsymbol{Q}_1 \text{diag}(1/\boldsymbol{g}) \boldsymbol{Q}_2^\intercal))^{1/p}$$
$$\text{s.t. } \boldsymbol{Q}_1 \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{g}), \boldsymbol{Q}_2 \in \Pi(\boldsymbol{\mu}_2, \boldsymbol{g}), \boldsymbol{g} \in \Delta_r$$

To solve the low-rank Gromov–Wasserstein problem in Eq. (8), (Scetbon et al., 2022) provides a mirror descent scheme. The general idea is to first solve the proximal point iteration under the KL divergence regularization $\boldsymbol{\xi}_i^{(t+1)} = \arg\min_{\boldsymbol{\xi}_i} \nabla_{\boldsymbol{\xi}_i}(\varepsilon_{\boldsymbol{A}_1, \boldsymbol{A}_2}(\boldsymbol{\xi}_1 \text{diag}(1/\boldsymbol{\xi}_3) \boldsymbol{\xi}_2^\intercal))^{1/p} + \text{KL}(\boldsymbol{\xi}_i, \boldsymbol{\xi}_i^{(t)})$ and project back to the feasible space $\mathcal{C} = \{(\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g}) | \text{s.t. } \boldsymbol{Q}_1 \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{g}), \boldsymbol{Q}_2 \in \Pi(\boldsymbol{\mu}_2, \boldsymbol{g}), \boldsymbol{g} \in \Delta_r\}$ using Dykstra's algorithm. We consider the most widely adopted 2-GW case in this paper, and provide the overall low-rank GW algorithm in Algorithm 3.

B.2. Variants of GeoMix

In the main content, we focus on the plain graphs w/o node attributes. when node attributes are available, we provide the following variant of GeoMix to incorporate node attributes in the mixup process. Specifically, when node attributes $\boldsymbol{X}_1 \in \mathbb{R}^{n_1 \times d}, \boldsymbol{X}_2 \in \mathbb{R}^{n_2 \times d}$ are available, we can calculate a cross-graph cost matrix $\boldsymbol{C} \in \mathbb{R}^{n_1 \times n_2}$, e.g., using Euclidean distance $\boldsymbol{C}(i, j) = \|\boldsymbol{X}_1(i, :) - \boldsymbol{X}_2(j, :)\|_2^2$. Similarly, following the mirror descent for solution, and each subproblem can

be written as follows:

$$
\begin{aligned}
&\left(\boldsymbol{Q}_1^{(t+1)}, \boldsymbol{Q}_2^{(t+1)}, \boldsymbol{g}^{(t+1)}\right) = \underset{\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g}}{\arg\min} \operatorname{KL}\left((\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{g}), (\boldsymbol{K}_1^{(t)}, \boldsymbol{K}_2^{(t)}, \boldsymbol{K}_3^{(t)})\right) \\
&\text{s.t. } \boldsymbol{Q}_1 \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{g}),\ \boldsymbol{Q}_2 \in \Pi(\boldsymbol{\mu}_2, \boldsymbol{g}),\ \boldsymbol{g} \in \Delta_r \\
&\text{where } \begin{cases}
\boldsymbol{K}_1^{(t)} = \exp\left(\gamma[-\alpha \boldsymbol{C} + 4(1-\alpha)\boldsymbol{A}_1 \boldsymbol{T}^{(t)} \boldsymbol{A}_2]\boldsymbol{Q}_2^{(t)} \operatorname{diag}(1/\boldsymbol{g}^{(t)}) + \log(\boldsymbol{Q}_1^{(t)})\right) \\
\boldsymbol{K}_2^{(t)} = \exp\left(\gamma[-\alpha \boldsymbol{C}^\intercal + 4(1-\alpha)\boldsymbol{A}_2 \boldsymbol{T}^{(t)^\intercal} \boldsymbol{A}_1]\boldsymbol{Q}_1^{(t)} \operatorname{diag}(1/\boldsymbol{g}^{(t)}) + \log(\boldsymbol{Q}_2^{(t)^\intercal})\right) \\
\boldsymbol{K}_3^{(t)} = \exp\left(\gamma \operatorname{diag}(\boldsymbol{Q}_1^{(t)^\intercal}[-\alpha \boldsymbol{C} + 4(1-\alpha)\boldsymbol{A}_2 \boldsymbol{T}^{(t)^\intercal} \boldsymbol{A}_1]\boldsymbol{Q}_2^{(t)})/\boldsymbol{g}^{(t)^2} + \log(\boldsymbol{g}^{(t)})\right) \\
\boldsymbol{T}^{(t)} = \boldsymbol{Q}_1^{(t)} \operatorname{diag}(1/\boldsymbol{g}^{(k)}) \boldsymbol{Q}_2^{(t)^\intercal}
\end{cases}
\end{aligned}
\tag{11}
$$

where $\alpha \in [0, 1]$ is the weight hyperparameter balancing the importance of structure and node attributes. We show that the computation for attributed and plain networks share the same big-O time complexity when adopting the Euclidean distance between node attributes as the cross-graph cost matrix, i.e., $\boldsymbol{C} = \boldsymbol{X}_1^2 \boldsymbol{1}_{d \times n_2} + \boldsymbol{1}_{n_1 \times d} \boldsymbol{X}_2^{2^\intercal} - 2\boldsymbol{X}_1 \boldsymbol{X}_2^\intercal$.

Compared to the computation for plain networks w/o node attributes, the computation in Eq. (11) involves three additional computations $\boldsymbol{C}\boldsymbol{Q}_2^{(t)} \operatorname{diag}(1/\boldsymbol{g}^{(t)})$, $\boldsymbol{C}^\intercal \boldsymbol{Q}_1^{(t)} \operatorname{diag}(1/\boldsymbol{g}^{(t)})$, and $\boldsymbol{Q}_1^{(t)^\intercal} \boldsymbol{C} \boldsymbol{Q}_2^{(t)} / \boldsymbol{g}_k^2$. Given the couplings $\boldsymbol{Q}_1^{(t)} \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{g}^{(t)}), \boldsymbol{Q}_2^{(t)} \in \Pi(\boldsymbol{\mu}_2, \boldsymbol{g}^{(t)})$ and simplex $\boldsymbol{g}^{(t)} \in \Delta_r$, we have $\boldsymbol{1}_{d \times n_1} \boldsymbol{Q}_1^{(t)} \operatorname{diag}(1/\boldsymbol{g}^{(t)}) = \boldsymbol{1}_{d \times n_2} \boldsymbol{Q}_2^{(t)} \operatorname{diag}(1/\boldsymbol{g}^{(t)}) = \boldsymbol{1}_{d \times r}$, we have the following time complexity analysis for calculating the additional three terms:

$$
\begin{aligned}
\boldsymbol{C}\boldsymbol{Q}_2^{(t)} \operatorname{diag}\left(1/\boldsymbol{g}^{(t)}\right) &= \underbrace{\boldsymbol{X}_1^2 \boldsymbol{1}_{d \times n_2}}_{\mathcal{O}(ndr)} + \underbrace{\boldsymbol{1}_{n_1 \times d} \boldsymbol{X}_2^{2^\intercal} \boldsymbol{Q}_2^{(t)} \operatorname{diag}\left(1/\boldsymbol{g}^{(t)}\right)}_{\mathcal{O}(ndr)} - \underbrace{2\boldsymbol{X}_1 \boldsymbol{X}_2^\intercal \boldsymbol{Q}_2^{(t)} \operatorname{diag}\left(1/\boldsymbol{g}^{(t)}\right)}_{\mathcal{O}(ndr)} \\
\boldsymbol{C}^\intercal \boldsymbol{Q}_1^{(t)} \operatorname{diag}\left(1/\boldsymbol{g}^{(t)}\right) &= \underbrace{\boldsymbol{1}_{n_2 \times d} \boldsymbol{X}_1^{2^\intercal} \boldsymbol{Q}_1^{(t)} \operatorname{diag}\left(1/\boldsymbol{g}^{(t)}\right)}_{\mathcal{O}(ndr)} + \underbrace{\boldsymbol{X}_2^{2^\intercal} \boldsymbol{1}_{d \times n_2}}_{\mathcal{O}(ndr)} - \underbrace{2\boldsymbol{X}_2^\intercal \boldsymbol{X}_1 \boldsymbol{Q}_1^{(t)} \operatorname{diag}\left(1/\boldsymbol{g}^{(t)}\right)}_{\mathcal{O}(ndr)} \\
\operatorname{diag}\left(\boldsymbol{Q}_1^{(t)^\intercal} \boldsymbol{C} \boldsymbol{Q}_2^{(t)}\right)/\boldsymbol{g}^{(t)^2} &= \underbrace{\operatorname{diag}\left(\boldsymbol{Q}_1^{(t)^\intercal} \boldsymbol{X}_1^2 \boldsymbol{1}_{d \times r} + \boldsymbol{1}_{r \times d} \boldsymbol{X}_2^{2^T} \boldsymbol{Q}_2^{(t)}\right)/\boldsymbol{g}^{(t)}}_{\mathcal{O}(ndr)} - \underbrace{2\operatorname{diag}\left(\boldsymbol{Q}_1^{(t)^\intercal} \boldsymbol{X}_1 \boldsymbol{X}_2^\intercal \boldsymbol{Q}_2\right)/\boldsymbol{g}^{(t)^2}}_{\mathcal{O}(nr^2)}
\end{aligned}
$$

Therefore, the additional time complexity for node attributes is $\mathcal{O}(KTndr)$, which is smaller than the $o(KTn^2r)$ of the original GEOMIX, i.e., considering node attributes does not carry additional time complexity in terms of the big-O notation.

C. Additional experiments

Sensitivity analysis of mixup ratio λ. We study the effect of mixup ratio λ on the model performance. The mixup ratio λ is randomly sampled from a uniform distribution $\text{Unif}(0, \lambda_{max})$, and we test how the model performance changes as λ_{max} changes from 0.05 to 0.95. Experiment results are shown in Figure 8. It is shown that GEOMIX is quite stable with different selections of λ.

Mixup graph visualization. We provide additional visualization for mixup graphs on real-world datasets MUTAG, IMDB-B, and IMDB-M in Figure 9. It is shown that $\tilde{\mathcal{G}}_{(\lambda)}$ smoothly transforms from $\mathcal{G}_1$ to $\mathcal{G}_2$ when λ increases, indicating that our mixup samples are consistent to their mixup labels, hence avoiding misleading supervision by suspicious sample–label pairs.

Besides, for two graphs $\mathcal{G}_1, \mathcal{G}_2$, the transformed graph $\Gamma_1(\mathcal{G}_1) = \tilde{\mathcal{G}}_{(0)}, \Gamma_2(\mathcal{G}_2) = \tilde{\mathcal{G}}_{(1)}$ share the same labels as $\mathcal{G}_1, \mathcal{G}_2$, i.e., $\tilde{\boldsymbol{y}}_{(0)} = \boldsymbol{y}_1, \tilde{\boldsymbol{y}}_{(1)} = \boldsymbol{y}_2$. According to sample–label consistency, , respectively. As shown in Figure 10, $\mathcal{G}_1, \mathcal{G}_2$ are quite similar to $\tilde{\mathcal{G}}_{(0.0)}, \tilde{\mathcal{G}}_{(1.0)}$, which validates that the employed transformations are equivalence-preserving and GEOMIX generate consistent sample–label pairs.

Embedding space visualization. We provide additional embedding space visualization for real-world datasets IMDB-M, PROTEINS, and MSRC-9 in Figure 10. These visualization results explain the experiment results in Table 1. For dataset

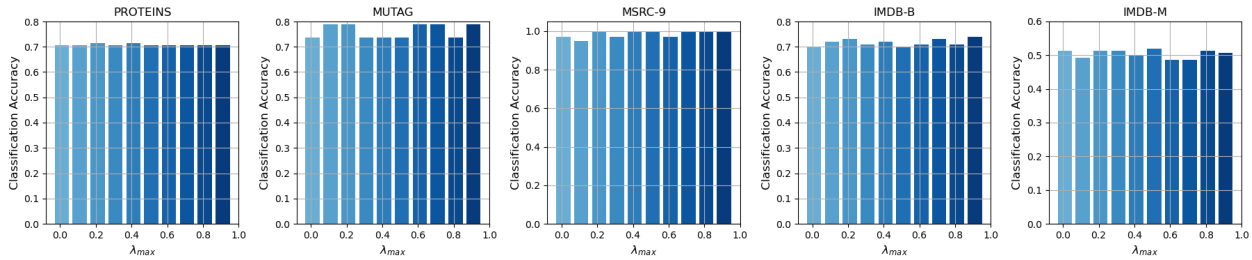


Figure 8. Sensitivity analysis on the mixup ratio. Mixup ratio λ is randomly sampled from Unif$(0, \lambda_{max})$, and we test λ_{max} ranging from 0.05 to 0.95.

like IMDB-B and IMDB-M, almost all of the mixup methods achieve limited improvement, as the data space has already been densely covered. But for datasets like MUTAG, PROTEINS, and MSRC-9 that are relatively sparsely covered, the mixup samples can siginicantly improve the model performance.

D. Reproducibility

Dataset description. All the real-world datasets used in the paper are from (Morris et al., 2020) and available online[1]. We give a brief introduction to the datasets as follows

- `PROTEINS` (Borgwardt et al., 2005) is a set of proteins where each graph represents the protein structure with nodes as amino acids and edges as chemical bonds. The graph labels indicate whether the proteins are enzymes or non-enzymes. The node labels correspond to atom type and the node attributes represent node chemical features.

- `MUTAG` (Debnath et al., 1991) is a set of chemical graphs, where each graph represents one nitro-aromatic compound with nodes as atoms and edges as chemical bonds. The binary graph labels indicate whether the compounds have mutagenicity on Salmonella typhimurium. The node labels correspond to the atom type.

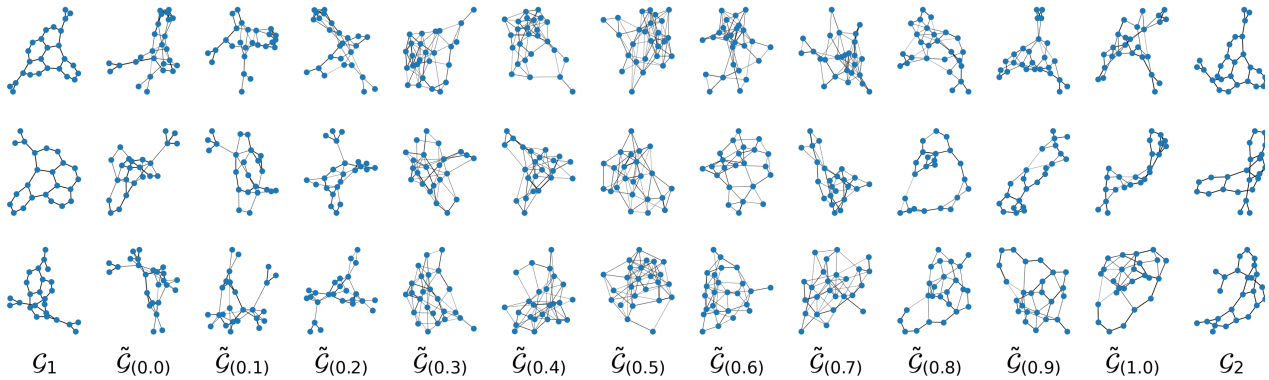
- `MSRC-9` (Neumann et al., 2016) is a set of semantic image networks, where each graph represents an image with nodes as superpixels and edges indicates the adjacent relationship between superpixels. The graph labels indicate the semantic meaning (e.g., building, grass, tree, etc.) of the image.

- `IMDB-B` and `IMDB-M` (Yanardag & Vishwanathan, 2015) are movie collaboration networks, where each graph represents one movie with nodes as actor/actress and edges indicating whether two actors/actresses co-appear in the same movie. The graph labels indicate the movie categories (Action/Romance for IMDB-B, and Comedy/Romance/Sci-Fi for IMDB-M).

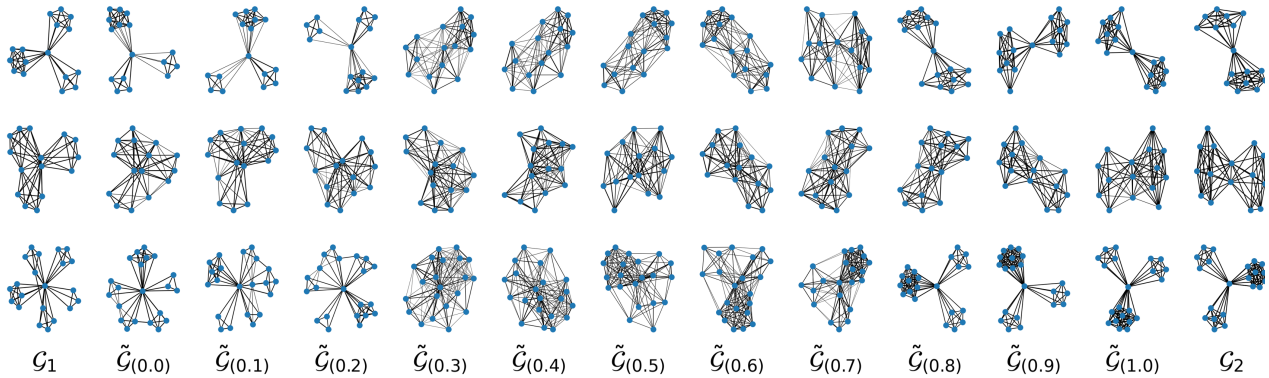
Table 4. Dataset statistics.

DATASET	#GRAPHS	#NODES	#EDGES	SPARSITY	#FEATURES	#GRAPH CLASS
PROTEINS	1,113	43.31	77.79	0.04	1	2
MUTAG	188	17.93	19.79	0.06	NONE	2
MSRC-9	221	40.58	97.94	0.06	NONE	8
IMDB-B	1,000	19.77	96.53	0.25	NONE	2
IMDB-M	1,500	12.74	53.88	0.33	NONE	3

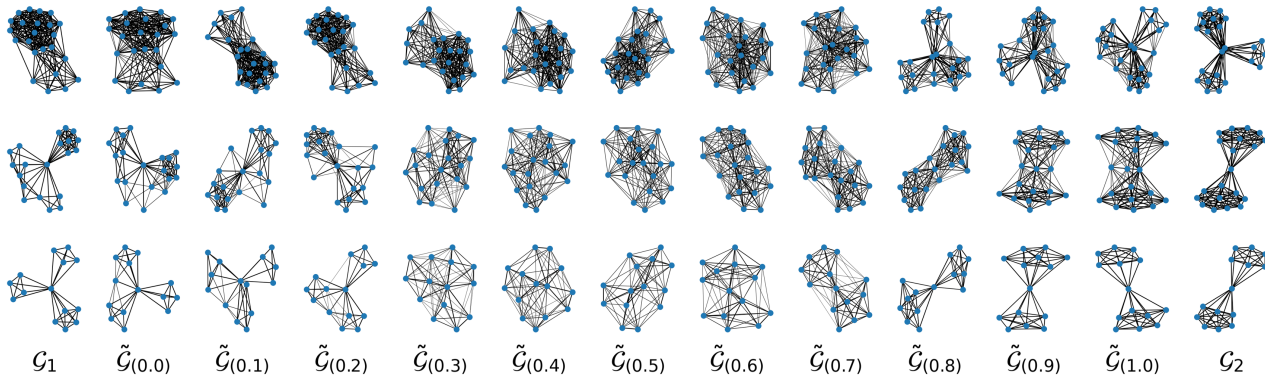
[1] https://chrsmrrs.github.io/datasets/



(a) MUTAG

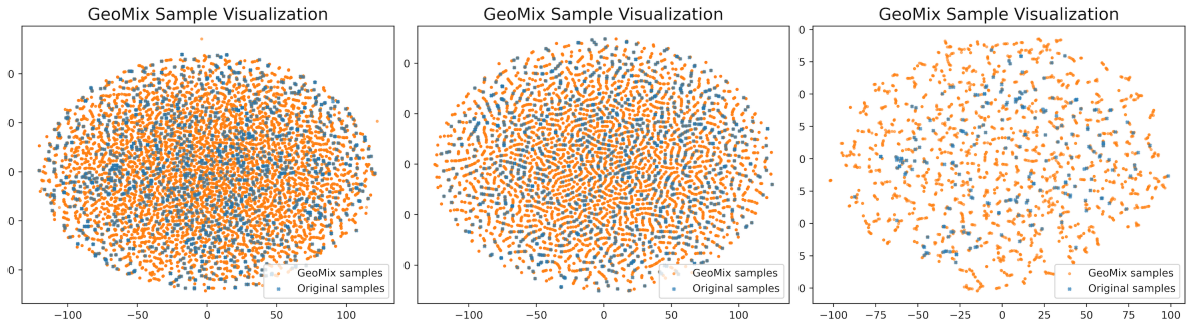


(b) IMDB-B



(c) IMDB-M

Figure 9. Visualization of mixup graphs on real-world datasets.



(a) IMDB-M (b) PROTEINS (c) MSRC-9

Figure 10. Visulization of the embedding space. The mixup samples lie on the geodesics connecting original samples, hence ensuring sample–label consistency.

Model architecture. For the GCN model (Kipf & Welling, 2017), we adopt three GCN layers with 32 hidden dimensions. We use ReLu as the activation function and global mean pooling as the readout. A dropout layer with dropout probability $p = 0.5$ is appended after the GCN layers, and followed by a linear layer with softmax activation to map embeddings to the classification probability.

For the GIN model (Xu et al., 2018), we adopt one 32-dimensional GIN layer with 3-layer MLP. We use ReLu as the activation function and global mean pooling as the readout. A dropout layer with dropout probability $p = 0.5$ is appended after the GIN layer, and followed by a linear layer with softmax activation to map embeddings to the classification probability.

Experiment pipeline. The proposed method is implemented in Python and all backbone models are built upon PyTorch. For model training, each model is trained for 300 epochs on the Linux platform with an Intel Xeon Gold 6240R CPU and an NVIDIA Tesla V100 SXM2 GPU.

During the experiments, for plain graphs without node attributes, we follow a standard procedure (Han et al., 2022) and use the categorized node degree as node attributes. We adopt 10-fold cross validation (train/test/validation=80%/10%/10%) to alleviate the randomness in the experiments.

E. Future Works and Limitations.

In this paper, we design a geodesic graph mixup framework to ensure sample–label consistency for graph-level mixup. In this section, we discuss possible directions and applications (Wei et al., 2023; 2024) to explore that can further benefit and extend the current framework, including:

- `Mixup sample selection.` It is essential to select good mixup samples to generate informative samples that can improve model generalization and robustness. According to (Zhang et al., 2018), mixup samples are expected to be far from the training examples to avoid memorization of the training samples. On possible solution is to adopt the GW distance as the graph distance and choose samples that are far away from other samples for mixup.
- `Node-level mixup.` In this paper, we focus on the graph classification task. It would be of great interest to generalize the framework to the node-level tasks such as node classification. For example, following the similar idea using $\boldsymbol{P}_1, \boldsymbol{P}_2$ to transform $\mathcal{G}_1, \mathcal{G}_2$ to well-aligned $\tilde{\mathcal{G}}_1, \tilde{\mathcal{G}}_2$, we can generate mixup node attributes as $(1-\lambda)\boldsymbol{P}_1^\top \boldsymbol{X}_1 + \lambda \boldsymbol{P}_2^\top \boldsymbol{X}_2$ and node labels as $(1-\lambda)\boldsymbol{P}_1^\top \boldsymbol{y}_1 + \lambda \boldsymbol{P}_2^\top \boldsymbol{y}_2$.