T-GAE: Transferable Graph Autoencoder for Network Alignment

Jiashu He
University of Pennsylvania
`jiashuhe@seas.upenn.edu`

Charilaos Kanatsoulis
Stanford University
`charilaos@cs.stanford.edu`

Alejandro Ribeiro
University of Pennsylvania
`aribeiro@seas.upenn.edu`

Abstract

Network alignment is the task of establishing one-to-one correspondences between the nodes of different graphs. Although finding a plethora of applications in high-impact domains, this task is known to be NP-hard in its general form. Existing optimization algorithms do not scale up as the size of the graphs increases. While being able to reduce the matching complexity, current GNN approaches fit a deep neural network on each graph and requires re-train on unseen samples, which is time and memory inefficient. To tackle both challenges we propose T-GAE, a transferable graph autoencoder framework that leverages transferability and stability of GNNs to achieve efficient network alignment on out-of-distribution graphs without retraining. We prove that GNN-generated embeddings can achieve more accurate alignment compared to classical spectral methods. Our experiments on real-world benchmarks demonstrate that T-GAE outperforms the state-of-the-art optimization method and the best GNN approach by up to 38.7% and 50.8%, respectively, while being able to reduce 90% of the training time when matching out-of-distribution large scale networks. We conduct ablation studies to highlight the effectiveness of the proposed encoder architecture and training objective in enhancing the expressiveness of GNNs to match perturbed graphs. T-GAE is also proved to be flexible to utilize matching algorithms of different complexities. Our code is available at `https://github.com/Jason-Tree/T-GAE`.

1 Introduction

Network alignment, also known as graph matching, is a classical problem in graph theory, that aims to find node correspondence across different graphs and is vital in a number of high-impact domains [Emmert-Streib et al., 2016]. In social networks, for instance, network alignment has been used for user deanonymization [Nilizadeh et al., 2014] and analysis [Ogaard et al., 2013], while in bioinformatics it is a key tool to identify functionalities in protein complexes [Singh et al., 2008], or to identify gene–drug modules [Chen et al., 2018a]. Graph matching also finds application in computer vision [Conte et al., 2003], sociology [Racz and Sridhar, 2021], to name a few. However, this problem is usually cast as a quadratic assignment problem (QAP), which is in general NP-hard.

Various approaches have been developed to tackle network alignment and can be divided into two main categories; i) optimization algorithms that attempt to approximate the QAP problem by relaxing the combinatorial constraints, ii) embedding methods that approach the problem by implicitly or explicitly generating powerful node embeddings that facilitate the alignment task. Optimization approaches, as [Anstreicher and Brixius, 2001, Vogelstein et al., 2015] employ quadratic programming relaxations, while [Klau, 2009] and [Peng et al., 2010] utilize semidefinite or Lagrangian-based relaxations respectively, [Du et al., 2019] and [Du et al., 2022] proposed to solve network alignment together with link prediction. Successive convex approximations were also proposed by [Konar and Sidiropoulos, 2020] to handle the QAP. Challenges associated with these methods include high computational cost, infeasible solutions, or nearly optimal initialization requirements. Embedding methods, on the other hand, overcome these challenges, but they usually produce inferior solutions, due to an inherent

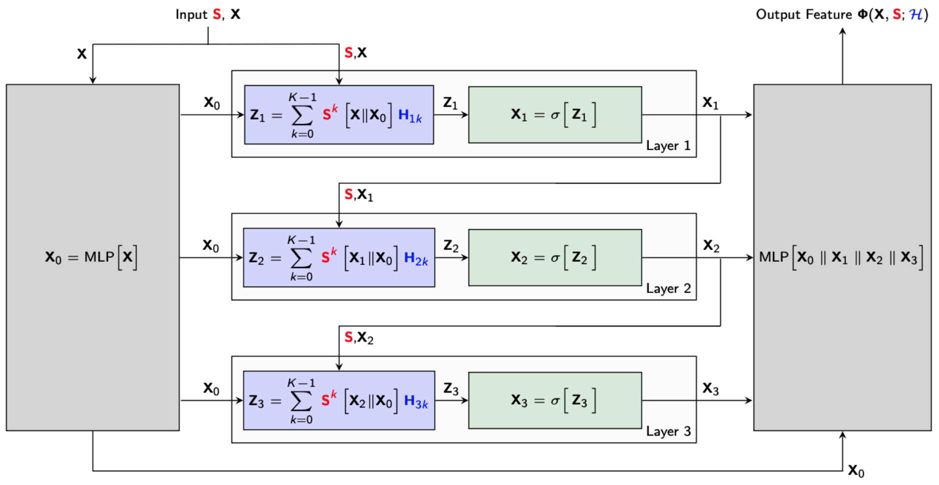
Input S, X
Output Feature $\Phi(X, S; \mathcal{H})$
$X_0 = \text{MLP}[X]$
$Z_1 = \sum_{k=0}^{K-1} S^k [X \| X_0] H_{1k}$
$X_1 = \sigma[Z_1]$
Layer 1
$Z_2 = \sum_{k=0}^{K-1} S^k [X_1 \| X_0] H_{2k}$
$X_2 = \sigma[Z_2]$
Layer 2
$Z_3 = \sum_{k=0}^{K-1} S^k [X_2 \| X_0] H_{3k}$
$X_3 = \sigma[Z_3]$
Layer 3
$\text{MLP}[X_0 \| X_1 \| X_2 \| X_3]$

Figure 1: To enhance the expressiveness of GNNs to align the unseen graphs, our proposed encoder processes the input features by a local MLP, we then incorporate attention mechanism on (1) input to each message-passing layer by attending to the output of the last layer and processed input feature. (2) output of the encoder by attending to the output of each message passing layer.

trade-off between embedding permutation-equivariance and the ability to capture the structural information of the graph. Typical embedding techniques include spectral and factorization methods [Umeyama, 1988, Feizi et al., 2019, Zhang and Tong, 2016, Kanatsoulis and Sidiropoulos, 2022], structural feature engineering methods [Berlingerio et al., 2013, Heimann et al., 2018], and random walk approaches [Perozzi et al., 2014, Grover and Leskovec, 2016a]. Recently [Chen et al., 2020, Karakasis et al., 2021] have proposed joint node embedding and network alignment, to overcome these challenges, but these methods do not scale up as the size of the graph increases.

Graph Neural Networks (GNNs) are powerful architectures that learn graph representations (embeddings) in a self-supervised way [Kipf and Welling, 2016, You et al., 2020]. They have shown state-of-the-art performance in several tasks, including biomedical [Gainza et al., 2020, Strokach et al., 2020, Jiang et al., 2021, Hu et al., 2023, Li et al., 2024], quantum chemistry [Gilmer et al., 2017], social networks and recommender systems [Ying et al., 2018, Wu et al., 2020, Liu et al., 2023a, Yang et al., 2024]. A line of studies has been conducted to formulate key-point matching on images as graph matching problems [Fey et al., 2020, Yu et al., 2020]. These frameworks rely on a powerful domain-specific encoder (CNNs, for example) to provide high-quality features. Given these high-quality features, GNNs are able to match graphs without training [Liu et al., 2022]. However, expressive features are expensive and sometimes infeasible to build [Zhang et al., 2020, Zhou et al., 2021]. [Liang et al., 2021] uses another trainable matrix to parameterize the internal connectivity between nodes of different graphs for faithful node alignment. Recently, [Gao et al., 2021] proposed use GNNs to learn node embedding, and match nodes with small Wasserstein distances. However, this method needs to fit a GNN on every input graph, which results in very high training cost, since training deep GNNs with large sizes graphs is computationally prohibitive, as GNNs have limited scalability with respect to graph and model sizes [Chen et al., 2018b,c, Chiang et al., 2019, Zeng et al., 2020].

To address these challenges, we propose T-GAE, a novel self-supervised GNN framework to perform network alignment. Specifically, we propose to utilize the transferability and robuseness of GNN to produce permutation equivariant and highly expressive embeddings. T-GAE trains the encoder on multiple families of small graphs and produce expressive/permutation equivariant representations for larger ***unseen*** networks. We further prove that GNN representations combine the eigenvectors of the graph in a nonlinear fashion and are at least as good in network alignment as certain spectral methods. T-GAE is a one-shot solution that tackles the challenges of real-time network alignment from (1) Optimization based algorithms: high computational cost, assume ground truth node correspondence as initialization. (2) Deep-learning based frameworks: re-train for every pair of graphs, rely on high quality of node features. Extensive experiments with real-world benchmarks demonstrate the effectiveness and efficiency of the proposed approach in both graph and sub-graph matching, thereby sheds light on the potential of GNN to tackle the highly-complex network optimization problems.

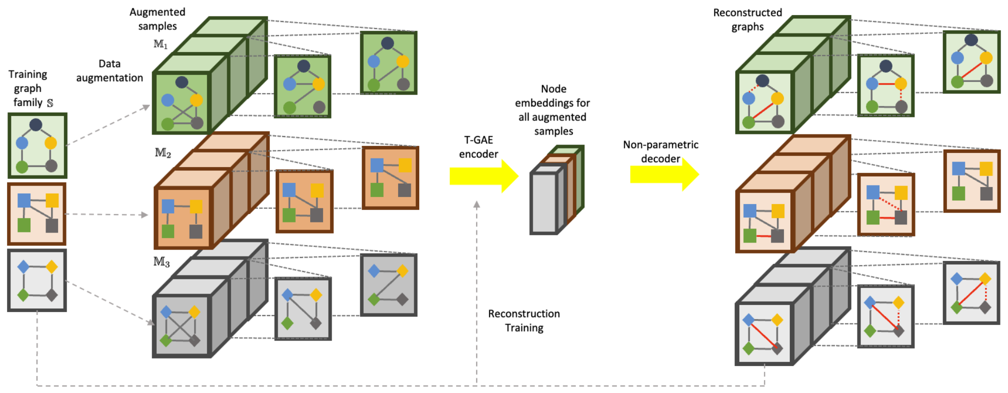

Figure 2: Proposed training objective: For each of the training graph, we generate a number of augmented samples by randomly adding or removing edges. The node embedding of these augmented samples are decoded non-parametrically, and compared with the corresponding original graph to train the T-GAE encoder.

2 Preliminaries

Graphs are represented by $\mathcal{G} := (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots, N\}$ is the set of vertices (nodes) and $\mathcal{E} = \{(v, u)\}$ correspond to edges between pairs of vertices. A graph is represented in a matrix form by a graph operator $\boldsymbol{S} \in \mathbb{R}^{N \times N}$, where $\boldsymbol{S}(i, j)$ quantifies the relation between node i and node j and $N = |\mathcal{V}|$ is the total number of vertices. In this work, we use the normalized graph adjacency and study the most general form of network alignment where there is no given graph attributes and ground truth node correspondence(anchor links).

2.1 Network Alignment

Definition 2.1 (Network Alignment)**.** Given a pair of graphs $\mathcal{G} := (\mathcal{V}, \mathcal{E})$, $\hat{\mathcal{G}} := (\hat{\mathcal{V}}, \hat{\mathcal{E}})$, with graph adjacencies $\boldsymbol{S}$, $\hat{\boldsymbol{S}}$, network alignment aims to find a bijection $g : \mathcal{V} \rightarrow \hat{\mathcal{V}}$ which minimizes the number of edge disagreements between the two graphs. Formally, the problem can be written as:

$$\min_{\boldsymbol{P} \in \mathcal{P}} \left\| \boldsymbol{S} - \boldsymbol{P} \hat{\boldsymbol{S}} \boldsymbol{P}^T \right\|_F^2, \tag{1}$$

where $\mathcal{P}$ is the set of permutation matrices.

As mentioned in the introduction, network alignment, is equivalent to the QAP, which has been proven to be NP-hard [Koopmans and Beckmann, 1957].

2.2 Spectral Decomposition of the Graph

A popular approach to tackle network alignment is by learning powerful node embeddings associated with connectivity information in the graph. Network alignment can be achieved by matching the node embeddings of different graphs rather than graph adjacencies, as follows:

$$\min_{\boldsymbol{P} \in \mathcal{P}} \left\| \boldsymbol{E} - \boldsymbol{P} \hat{\boldsymbol{E}} \right\|_F^2, \tag{2}$$

where $\boldsymbol{E} \in \mathbb{R}^{N \times F}$ is embedding matrix and $\boldsymbol{E}[i, :]$ is the vector representation of node i. The optimization problem in (2) is a linear assignment problem and can be optimally solved in $\mathcal{O}\left(N^3\right)$ by the Hungarian method [Kuhn, 1955a]. Simpler sub-optimal alternatives also exist that operate with $\mathcal{O}\left(N^2\right)$ or $\mathcal{O}\left(N \log(N)\right)$ flops.

A question that naturally arises is how to generate powerful node embeddings that capture the network connectivity and also be effective in aligning different graphs. A natural and effective approach is to leverage the spectral decomposition of the graph, $\boldsymbol{S} = \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^T$, where $\boldsymbol{V}$ is the orthonormal matrix of the eigenvectors, and $\boldsymbol{\Lambda}$ is the diagonal matrix of corresponding eigenvalues. Note that we assume undirected graphs and thus $\boldsymbol{S}$ is symmetric. Spectral decomposition has been proven to be an efficient approach to generating meaningful node embedding for graph matching [Umeyama, 1988, Feizi et al., 2019]. In particular, $\boldsymbol{E} = \boldsymbol{V}$ or $\boldsymbol{E} = \boldsymbol{V\Lambda}$ are node embeddings that capture the network connectivity since they can perfectly reconstruct the graph. However, $\boldsymbol{V}$ is not unique. Thus computing the spectral decomposition of the same graph with node relabelling, $\tilde{\boldsymbol{S}} = \boldsymbol{PSP}^T$ is not guaranteed to produce a permuted version of $\boldsymbol{V}$, i.e., $\boldsymbol{PV}$. Even in the case where $\boldsymbol{S}$ does not have repeated eigenvalues $\boldsymbol{V}$ is only unique up to column sign, which prevents effective matching.

To overcome the aforementioned uniqueness limitation, one can focus on the top m eigenvectors that correspond to non-repeated eigenvalues in both $\boldsymbol{S}$ and $\hat{\boldsymbol{S}}$ and compute their absolute values. Then network alignment can be cast as:

$$\min_{\boldsymbol{P}\in\mathcal{P}} \left\| \left|\boldsymbol{V}_m\right| - \boldsymbol{P}\left|\hat{\boldsymbol{V}}_m\right| \right\|_F^2, \tag{3}$$

where $\boldsymbol{V}_m \in \mathbb{R}^{N\times m}$ corresponds to the subspace of non-repeated eigenvalues. The formulation in (3) is a similar to the problem solved in [Umeyama, 1988].

3 Graph Neural Networks (GNNs) Upper-Bounds Spectral Methods for Network Alignment

A GNN is a cascade of layers and performs local, message-passing operations that are usually defined by the following recursive equation:

$$x_v^{(l+1)} = g\left(x_v^{(l)}, f\left(\left\{x_u^{(l)} : u \in \mathcal{N}\left(v\right)\right\}\right)\right), \tag{4}$$

where $\mathcal{N}\left(v\right)$ is the neighborhood of vertex v, i.e., $u \in \mathcal{N}\left(v\right)$ if and only if $(u, v) \in \mathcal{E}$. The function f operates on multisets ($\{\cdot\}$ represents a multiset) and $f,\ g$ are ideally injective. Common choices for f are the summation or mean function, and for g the linear function, or the multi-layer perceptron (MLP).

Overall, the output of the $L-$th layer of a GNN is a function $\phi\left(\boldsymbol{X}; \boldsymbol{S}, \mathcal{H}\right) : \mathbb{R}^{N\times D} \rightarrow \mathbb{R}^{N\times D_L}$, where $\boldsymbol{S}$ is the graph operator, and $\mathcal{H}$ is the tensor of the trainable parameters in all L layers and produces D_L- dimensional embeddings for the nodes of the graph defined by $\boldsymbol{S}$.

GNNs admit some very valuable properties. First, they are permutation equivariant:

Theorem 3.1 ([Xu et al., 2019a, Maron et al., 2018])**.** *Let $\phi\left(\boldsymbol{X}; \boldsymbol{S}, \mathcal{H}\right) : \mathbb{R}^{N\times D} \rightarrow \mathbb{R}^{N\times D^L}$ be a GNN with parameters $\mathcal{H}$. For $\tilde{\boldsymbol{X}} = \boldsymbol{PX}$ and $\tilde{\boldsymbol{S}} = \boldsymbol{PSP}^T$ that correspond to node relabelling according to the permutation matrix $\boldsymbol{P}$, the output of the GNN takes the form:*

$$\tilde{\boldsymbol{X}}^{(L)} = \phi\left(\tilde{\boldsymbol{X}}; \tilde{\boldsymbol{S}}, \mathcal{H}\right) = \boldsymbol{P}\phi\left(\boldsymbol{X}; \boldsymbol{S}, \mathcal{H}\right) \tag{5}$$

The above property is not satisfied by other spectral methods. GNNs are also stable [Gama et al., 2020, Parada-Mayorga et al., 2023, Ruiz et al., 2021], transferable [Ruiz et al., 2020], and have high expressive power [Xu et al., 2019a, Abboud et al., 2021, Kanatsoulis and Ribeiro, 2022].

3.1 GNNs and Network Alignment

To characterize the ability of a GNN to perform network alignment we first pointed out the GNNs perform nonlinear spectral operations. Details can be found in Appendix Section D.1. We can further prove that:

Theorem 3.2. *Let $\mathcal{G},\ \hat{\mathcal{G}}$ be graphs with adjacencies $\boldsymbol{S},\ \hat{\boldsymbol{S}}$ that have non-repeated eigenvalues. Also let $\boldsymbol{P}^\diamond,\ \check{\boldsymbol{P}}$ be solutions to the optimization problems in (1) and (3) respectively. Then there exists a GNN $\phi\left(\boldsymbol{X}; \boldsymbol{S}, \mathcal{H}\right) : \mathbb{R}^{N\times D} \rightarrow \mathbb{R}^{N\times D^L}$ such that:*

$$\left\| \boldsymbol{S} - \boldsymbol{P}^\diamond \hat{\boldsymbol{S}} \boldsymbol{P}^{\diamond^T} \right\|_F^2 \leq \left\| \boldsymbol{S} - \boldsymbol{P}^* \hat{\boldsymbol{S}} \boldsymbol{P}^{*^T} \right\|_F^2 \leq \left\| \boldsymbol{S} - \check{\boldsymbol{P}} \hat{\boldsymbol{S}} \check{\boldsymbol{P}}^T \right\|_F^2$$

with

$$\boldsymbol{P}^* = \arg\min_{\boldsymbol{P}\in\mathcal{P}} \left\| \phi\left(\boldsymbol{X}; \boldsymbol{S}, \mathcal{H}\right) - \boldsymbol{P}\phi\left(\hat{\boldsymbol{X}}; \hat{\boldsymbol{S}}, \mathcal{H}\right) \right\|_F^2.$$

The proof can be found in Appendix D. The assumption that the graph adjacencies have different eigenvalues is not restrictive. Real nonisomorphic graphs have different eigenvalues with very high probability [Haemers and Spence, 2004]. Theorem 3.2 compares the network alignment power of a GNN with that of a spectral algorithm Umeyama [1988], that uses the absolute values of graph adjacency eigenvectors to match two different graphs. According to Theorem 3.2 there always exists a GNN that can perform at least as well as the spectral approach. The proof studies a GNN with white random input and measures the variance of the filter output. Then it shows that message-passing layers are able to compute the absolute values of the graph adjacency eigenvectors when the adjacency has non-repeated eigenvalues. As a result there always exists a single layer GNN that outputs the same node features as the ones used in Umeyama [1988], which concludes our proof. The questions is: How do we train such a GNN that is (1) Expressive to the structural information thus its output can be used to match corresponding nodes. (2) Robust to different perturbations and even larger scale unseen graphs so that it can be deployed efficiently without re-training. (3) Agnostic to the ground truth node correspondence so that it can be trained unsupervisedly, which makes it generalizable to most real-world settings. To answer these questions, we introduce our proposed training framework in the following section.

4 Proposed Method

We now leverage the favorable properties of GNNs (permutation equivariance, expressivity, and transferability) to tackle real-world network alignment. Our approach learns low-dimensional node embeddings (Eq. 4) that enable graph matching via solving the linear assignment in (2) rather than a quadratic assignment problem in (1). We design a robust GNN framework such that the node embeddings are expressive to accurately match similar nodes and also stable to graph perturbations.

4.1 Learning Network Geometry with Transferable Graph Auto-encoders

The goal of the proposed framework is to learn a function that maps graphs to node representations and effectively match nodes from different graphs. This function is modeled by a GNN encoder $\phi\left(\boldsymbol{X};\boldsymbol{S},\mathcal{H}\right)$, described by Fig. 1. The learned encoder should work for a family of training graphs $\{\mathcal{G}_0,\ldots,\mathcal{G}_i,\ldots,\mathcal{G}_I\}$ with a set of adjacency matrices $\mathbb{S}=\{\boldsymbol{S}_0,\ldots,\boldsymbol{S}_i,\ldots,\boldsymbol{S}_I\}$, rather than a single graph. So the idea is not to train a GNN on a single graph [Kipf and Welling, 2016], but train a transferable graph auto-encoder by solving the following optimization problem.

$$\min_{\mathcal{H}} \mathbb{E}\left[l\left(\rho\left(\phi\left(\boldsymbol{X};\boldsymbol{S}_i,\mathcal{H}\right)\phi\left(\boldsymbol{X};\boldsymbol{S}_i,\mathcal{H}\right)^T\right),\boldsymbol{S}_i\right)\right], \tag{6}$$

where $l\left(\cdot\right)$ is the binary cross entropy (BCE) and $\rho\left(\cdot\right)$ is the logistic function. $\boldsymbol{S}_i\in\mathbb{S}$ is a realization from a family of graphs and the expectation (empirical expectation is practice) is computed over this graph family. The generalized framework in (6) learns a mapping from graphs to node representations, and can be applied to ***out-of-distribution*** graphs that have not been observed during training. This twist in the architecture enables node embedding and graph matching for the unseen and larger scale networks without re-training, where fitting a GNN is computationally prohibitive in real-world applications.

4.2 Robust and Generalizable Node representations with self-supervised learning (data augmentation)

So far we proposed a GNN framework to produce expressive node representations to perform network alignment. In this subsection, we further upgrade our framework by ensuring the robustness and generalization ability of the proposed mapping. In particular, for each graph, $\boldsymbol{S}_i\in\mathbb{S}$, we augment the training set with perturbed versions that are described by the following set of graph adjacencies $\mathbb{M}_i=\left\{\boldsymbol{S}_i^{(0)},\ldots,\boldsymbol{S}_i^{(j)},\ldots,\boldsymbol{S}_i^{(J)}\right\}$, that are perturbed versions of $\boldsymbol{S}_i$. To do so we add or remove an edge with a certain probability yielding $\tilde{\boldsymbol{S}}_i\in\mathbb{M}$, such that $\tilde{\boldsymbol{S}}_i=\boldsymbol{S}_i+\boldsymbol{M}_i$, where $\boldsymbol{M}_i\in\{-1,0,1\}^{N\times N}$. Note that $\boldsymbol{M}$ changes for each $\tilde{\boldsymbol{S}}_i$, and $\boldsymbol{M}[m,n]$ can be equal to 1 and -1 only if $\boldsymbol{S}[m,n]$ is equal to 0 and 1 respectively. To train the proposed transferable graph-autoencoder we consider the following optimization problem:

$$\min_{\mathcal{H}} \mathbb{E}_{\mathbb{S}}\left[\mathbb{E}_{\mathbb{M}_i}\left[l\left(\rho\left(\phi\left(\boldsymbol{X};\tilde{\boldsymbol{S}}_i,\mathcal{H}\right)\phi\left(\boldsymbol{X};\tilde{\boldsymbol{S}}_i,\mathcal{H}\right)^T\right),\boldsymbol{S}_i\right)\right]\right], \tag{7}$$

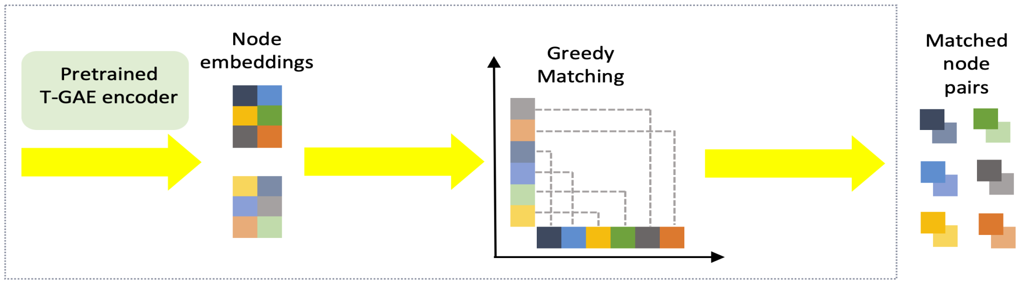

Figure 3: The pre-trained encoder operates on out-of-distribution samples. The generated node embeddings are then matched greedily.

where $\mathbb{E}_{\mathbb{S}}$ is the expectation with respect to the family of graphs $\mathbb{S}$ and $\mathbb{E}_{\mathbb{M}_i}$ is the expectation with respect to the perturbed graphs $\mathbb{M}_i$. In practice, $\mathbb{E}_{\mathbb{S}}$, $\mathbb{E}_{\mathbb{M}}$ correspond to empirical expectations. Note that training according to (7) also benefits the robustness of the model, which is crucial in deep learning tasks [Wang et al., 2022, He, 2021]. A schematic illustration of the training process can be found in Fig. 2.

Remark 4.1. (Large-scale network alignment by transferability of GNN)

The proposed framework learns a mapping $\phi : \mathbb{G} \to \mathbb{R}^{N \times F}$ that produces expressive and robust node representations for a family of graphs $\mathcal{G} \in \mathbb{G}$. This mapping is designed in such a way that the problem in (2) approximates the problem in (1) and allows solving network alignment in polynomial time. One of the main benefits of the proposed framework is that it enables large-scale network alignment. Task specific augmentation during training is the key to prompt transferability of deep neural networks [Li et al., 2022]. And the transferability analysis of GNN encoders [Ruiz et al., 2020] suggests that we can train with small graphs and efficiently execute with much larger graphs when the substructures (motifs) that appear in the tested graphs, were also partially observed during training. Since the proposed transferable graph auto-encoder is trained with multiple graphs, a variety of motifs are observed during training, which cannot be observed with a classical graph autoencoder, and the proposed GNN encoder can be transferred to larger-scale out-of-distribution graphs.

4.3 Alignment and Complexity analysis

After learning the powerful node embeddings, network alignment is performed by solving the linear assignment problem in (2). An illustration of the assignment is presented in Fig. 3. The node features produced by T-GAE are used to calculate a pairwise distance matrix, followed by the greedy Hungarian algo rithm to predict node correspondences. To analyze the complexity of our approach we study the 3 main parts of T-GAE: a) The design of the input structural features, b) The message-passing GNN that produces node embeddings, and c) the linear assignment algorithm.

The computation of our neighborhood-based structural features is expected to take $\mathcal{O}\left(|\mathcal{V}|\right)$ in real graphs, as proved in Henderson et al. [2011]. The computational and memory complexity of the message-passing GNN is $\mathcal{O}\left(|\mathcal{V}|\, c^2 + |\mathcal{E}|\, c\right)$, and $\mathcal{O}\left(|\mathcal{V}|\, c\right)$, where c is the width of the GNN. The computational complexity to align the nodes of the graph is $\mathcal{O}\left(|\mathcal{V}|^2\right)$ since we are using the suboptimal greedy Hungarian. If we want to optimally solve the linear assignment problem we need to use the Hungarian algorithm that has $\mathcal{O}\left(|\mathcal{V}|^3\right)$ complexity. If we want to process large graphs we can use efficient nearest neighbors algorithms with complexity $\mathcal{O}\left(|\mathcal{V}| \log\left(|\mathcal{V}|\right)\right)$ to perform soft linear assignment. However, this efficient algorithm only works to match graphs with its permuted samples. We include detailed discussion in Appendix Section H. Overall the complexity of T-GAE is $\mathcal{O}\left(|\mathcal{V}|^2\right)$, or $\mathcal{O}\left(|\mathcal{V}|\, c^2 + |\mathcal{E}|\, c + |\mathcal{V}| \log\left(|\mathcal{V}|\right)\right)$ for un-perturbed samples.

5 Experiments

5.1 Experiments Setup

The experiments included in this section are designed to answer the following research questions:(1) Can `T-GAE` generate competing graph matching accuracy on real-world networks from various domains with different sizes, while being efficient by utilizing transferability of GNN for large-scale our-of-distribution graphs? We answer this question in Section 5 and Appendix Section G by comparing the performance on matching small to middle sized graphs with unseen perturbed samples, and large scale out-of-distribution networks under different perturbation distributions. (2) Is `T-GAE` robust to different graph matching tasks and real-world noise distributions? In Section 5.3, we conduct sub-graph matching experiments to align two different real-world networks that are partially aligned. (3) How does the proposed network architecture (Figure 1) and the training objective (Figure 2) contribute to network alignment? To demonstrate the contribution of each proposed component, we conduct ablation studies in Section 5.4, to compare T-GAE with untrained T-GAE, T-GAE trained with GAE objective, and GAE. (4) How much efficiency does T-GAE offer compared to the existing optimization and GNN approaches, and what are the possible trade-offs between efficiency and matching accuracy of T-GAE? We compare the efficiency of different matching algorithms and empirically prove that (a) The matching accuracy of T-GAE can be further improved by leveraging the exact Hungarian algorithm. (b) The efficiency of T-GAE to match graphs with their permuted samples can be enhanced if we adopt the more efficient matching algorithm introduced in Section 4.3. This set of experiments are included in Appendix Section H.

For each of the above mentioned experiments, we compare T-GAE with three categories of graph matching approaches: (a)GNN based methods: `WAlign` [Gao et al., 2021], `GAE` and `VGAE` [Kipf and Welling, 2016]; (b)Graph/Node embedding techniques: `NetSimile` [Berlingerio et al., 2013], `Spectral` [Umeyama, 1988], `DeepWalk` [Perozzi et al., 2014], [Grover and Leskovec, 2016b], `GraphWave` [Donnat et al., 2018] and `LINE` [Tang et al., 2015]. (c)Optimization based graph matching algorithms: `S-GWL` [Xu et al., 2019b], `ConeAlign` [Chen et al., 2020] and `FINAL` [Zhang and Tong, 2016]. Note that `LINE, VGAE, DeepWalk`, and `Node2Vec` are omitted from some experiments since they show very poor performance. The reason behind that is that they are not permutation equivariant. GraphWave is also excluded from the sub-graph matching experiment, it could not identify correlated nodes in two different graphs. In the case of graphs without attributes `FINAL` is equivalent to the popular `Isorank` [Singh et al., 2008] algorithm, and `FINAL` is omitted in sub-graph matching experiments due to weak performance.

We include detailed descriptions of our included datasets, implementation details of T-GAE and all the competing baselines in Appendix Section E.

5.2 Graph Matching Experiments

In this subsection we compare the performance of T-GAE with all competing baselines to match the graphs with permuted and perturbed versions of them. In particular, let $\mathcal{G}$ be a graph with adjacency matrix $\boldsymbol{S}$. We then produce 10 permuted-perturbed versions according to $\hat{\boldsymbol{S}} = \boldsymbol{P}\left(\boldsymbol{S}+\boldsymbol{M}\right)\boldsymbol{P}^T$, where $\boldsymbol{M} \in \{-1,0,1\}^{N\times N}$ and $\boldsymbol{P}$ is a permutation matrix. For each perturbation level $p \in \{0, 1\%, 5\%\}$, the total number of perturbations is defined as $p|\mathcal{E}|$, where $|\mathcal{E}|$ is the number of edges of the original graph.

Specifically, we train `T-GAE` according to (7), where $\mathbb{S}$ consist of the small-size networks, i.e., Celegans, Arena, Douban, and Cora. Then we resort to transfer learning and use the `T-GAE` encoder to produce node embedding for perturbed versions of (a) Celegans, Arena, Douban, and Cora, and (b) larger graphs, i.e., Dblp, and Coauthor CS. Note that none of these perturbed versions were considered during training. This is in contrast with all GNN baselines that are retrained on every pair of networks in the testing dataset. We report the average and standard deviation of the matching accuracy for 10 randomly generated perturbation samples under uniform edge editing in Table 1, where each edge and non-edge shares the same probability of being removed or augmented. We report the results for removing edges according to degrees and the relevant discussed in Appendix G.

T-GAE framework results in a robust and transferable GNN to perform network alignment at a large scale. Our first observation is that for zero perturbation most algorithms are able to achieve a high level of matching accuracy. This is expected, since for zero perturbation the network alignment is equivalent to graph isomorphism. On the smaller networks(Celegans, Arenas, Douban, Cora), `T-GAE` performs at least as well as the current state-of-the-art optimization approaches (`S-GWL` and `ConeAlign`). Specifically, it achieves up to 38.7% and 44.7% accuracy increase compared to `S-GWL` and `ConeAlign`, respectively. Regarding the ability of `T-GAE` to perform large-scale network alignment the results are definitive. `T-GAE` enables low-complexity training with small graphs, and execution at larger settings by leveraging transfer learning, and it consistently outperforms all

Table 1: Graph matching accuracy on 10 randomly perturbed samples under different levels of edge editing on Uniform model. The proposed T-GAE is trained on the clean Celegans, Arena, Douban, and Cora networks, and tested on noisy versions of them and the larger Dblp, and Coauthor CS. Accuracy above 80% is highlighted in green, 60% to 80% accuracy is in yellow, and performance below 60% is in red.

	Dataset \ Algorithm	*Feature Engineering based*			*Optimization based*			*GNN based*		
		Spectral	Netsimile	GraphWave	FINAL	S-GWL	ConeAlign	WAlign	GAE	**T-GAE**
0% perturbation	Celegans	87.8 ± 1.5	72.7 ± 0.9	65.3 ± 1.7	92.2 ± 1.2	$\mathbf{93.0 \pm 1.5}$	66.6 ± 1.2	88.4 ± 1.6	86.3 ± 1.3	91.0 ± 1.1
	Arenas	97.7 ± 0.4	94.7 ± 0.3	81.7 ± 0.7	97.5 ± 0.3	97.5 ± 0.3	87.8 ± 0.6	97.4 ± 0.5	97.6 ± 0.4	$\mathbf{97.8 \pm 0.4}$
	Cora	85.0 ± 0.4	73.7 ± 0.4	8.3 ± 0.4	87.5 ± 0.7	87.3 ± 0.7	38.5 ± 0.7	87.2 ± 0.4	87.1 ± 0.8	$\mathbf{87.5 \pm 0.4}$
	Douban	89.9 ± 0.4	46.4 ± 0.4	17.5 ± 0.2	89.9 ± 0.3	$\mathbf{90.1 \pm 0.3}$	68.1 ± 0.4	90.0 ± 0.4	89.5 ± 0.4	$\mathbf{90.1 \pm 0.3}$
	Dblp	84.5 ± 0.1	63.7 ± 0.2	doesn't scale	$\mathbf{85.6 \pm 0.2}$	> 48 hours	44.3 ± 0.6	85.6 ± 0.2	85.2 ± 0.3	$\mathbf{85.6 \pm 0.2}$
	Coauthor CS	97.5 ± 0.1	90.9 ± 0.1	doesn't scale	$\mathbf{97.6 \pm 0.1}$	> 48 hours	75.8 ± 0.5	97.5 ± 0.2	97.6 ± 0.3	$\mathbf{97.6 \pm 0.1}$
1% perturbation	Celegans	68.5 ± 16.1	66.3 ± 3.8	22.5 ± 22.4	33.2 ± 7.8	$\mathbf{87.1 \pm 6.1}$	60.9 ± 2.5	80.7 ± 3.0	33.2 ± 8.4	86.5 ± 1.1
	Arenas	85.0 ± 10.0	87.8 ± 1.0	40.5 ± 23.8	32.5 ± 5.9	94.2 ± 0.7	84.6 ± 1.0	90.0 ± 3.1	30.1 ± 17.6	$\mathbf{96.0 \pm 1.0}$
	Cora	59.1 ± 9.3	66.4 ± 1.6	3.7 ± 2.9	30.0 ± 3.3	46.4 ± 6.9	33.5 ± 1.6	80.1 ± 1.2	57.9 ± 5.3	$\mathbf{85.1 \pm 0.5}$
	Douban	25.8 ± 27.2	40.0 ± 1.2	9.9 ± 5.9	27.8 ± 5.7	72.1 ± 0.7	64.7 ± 0.4	77.2 ± 4.8	38.3 ± 16.4	$\mathbf{87.3 \pm 0.4}$
	Dblp	55.6 ± 19.0	55.1 ± 1.7	doesn't scale	15.2 ± 3.3	> 48 hours	37.8 ± 1.1	73.1 ± 1.6	19.4 ± 0.6	$\mathbf{83.3 \pm 0.4}$
	Coauthor CS	58.2 ± 22.1	75.2 ± 2.2	doesn't scale	13.3 ± 5.0	> 48 hours	68.5 ± 2.8	75.2 ± 5.4	49.5 ± 7.8	$\mathbf{93.2 \pm 0.8}$
5% perturbation	Celegans	24.9 ± 15.9	41.1 ± 13.0	7.6 ± 9.2	10.4 ± 2.7	68.3 ± 12.7	50.5 ± 3.4	42.4 ± 21.1	6.5 ± 2.4	$\mathbf{69.2 \pm 2.1}$
	Arenas	52.1 ± 16.5	52.3 ± 5.3	6.9 ± 7.2	7.2 ± 2.6	$\mathbf{88.3 \pm 3.2}$	75.0 ± 2.7	30.4 ± 17.5	1.4 ± 1.4	81.2 ± 1.4
	Cora	29.5 ± 0.8	41.2 ± 3.3	0.8 ± 0.3	6.7 ± 2.8	39.9 ± 5.5	23.0 ± 2.0	33.4 ± 7.3	9.6 ± 2.7	$\mathbf{67.7 \pm 1.3}$
	Douban	23.8 ± 20.6	20.7 ± 4.6	1.9 ± 2.8	7.8 ± 3.0	68.6 ± 0.8	54.1 ± 1.2	36.6 ± 13.4	0.6 ± 0.3	$\mathbf{70.2 \pm 2.5}$
	Dblp	28.0 ± 7.8	19.5 ± 4.8	doesn't scale	2.7 ± 0.9	> 48 hours	24.4 ± 2.9	15.9 ± 8.3	1.4 ± 0.2	$\mathbf{60.8 \pm 1.9}$
	Coauthor CS	9.7 ± 5.0	26.3 ± 6.0	doesn't scale	2.0 ± 0.4	> 48 hours	51.4 ± 5.1	11.3 ± 7.5	0.6 ± 0.1	$\mathbf{66.0 \pm 1.4}$

Table 2: Subgraph matching performance comparison. The proposed T-GAE is trained on the two real-world graphs, and test to match the aligned portion of them.

Algorithm \ Hit Rate	*ACM-DBLP*				*Douban Online-Offline*			
	Hit@1	Hit@5	Hit@10	Hit@50	Hit@1	Hit@5	Hit@10	Hit@50
Netsimile	2.59%	8.32%	12.09%	26.42%	1.07%	2.77%	4.74%	15.03%
Spectral	1.40%	4.62%	7.21%	16.34%	0.54%	1.34%	2.95%	13.95%
GAE	8.1%	22.5%	30.1%	45.1%	3.3%	9.2%	14.1%	32.1%
WAlign	62.02%	81.96%	87.31%	93.89%	36.40%	53.94%	67.08%	85.33%
T-GAE	**73.89%**	**91.73%**	**95.33%**	**98.22%**	**36.94%**	**60.64%**	**69.77%**	**89.62%**

competing baselines on the two networks with more than 10k nodes. In particular, it is able to achieve very high levels of matching accuracy for both Dblp and Coauthor CS, for $p = 0\%,\ 1\%$. It is also the only method that consistently achieves at least 60% accuracy at 5% perturbation. To the best of our knowledge, our experiments on DBLP [Pan et al., 2016] and Coauthor CS [Shchur et al., 2018] are the first attempts to perform exact alignment on networks at the order of 20k nodes and 80k edges.

The benefits of processing structural node features with T-GAE is clear. There is a clear benefit of processing the structural embeddings with `TGAE` since it offers up to 43.7% performance increase compared to `NetSimile`. When some perturbation is added, the conclusions are straightforward. Especially when perturbations are added, our proposed `T-GAE` markedly outperforms all the competing GNN alternatives(`WAlign` and `GAE`) and shows the desired robustness to efficiently perform network alignment. We observe that neither GAE nor WAlign is robust to noise. This highlights the benefit of T-GAE in handling the distribution shift brought by the structural dissimilarity between different graphs.

5.3 Sub-graph Matching Experiments

We further test the performance of T-GAE in matching subgraphs of different networks that have aligned nodes (nodes that represent the same entities in different networks). Specifically, in ACM-DBLP, the task is to match the papers that appear in both citation networks; in Douban Online-Offline, we aim to identify the users that take part into both online and offline activities. Note that this is the most realistic graph matching experiment we can perform since we match 2 different real graphs with partially aligned nodes.

T-GAE uniformly achieves the best performance in the most realistic scenario of network alignment. Most optimization based approaches do not generalize to this real-world scenario because their optimization objective usually prevents from matching graphs with different numbers of nodes. There is a significant improvement in matching accuracy with GNN-based methods

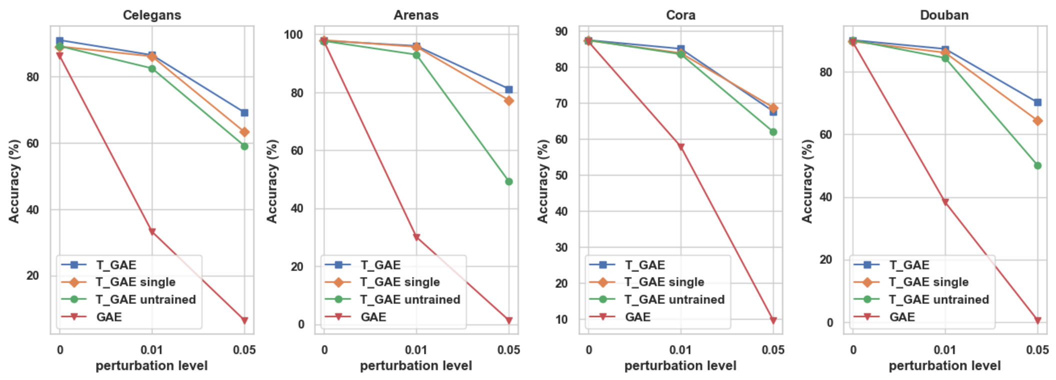

Figure 4: Graph matching performance comparison of T-GAE,T-GAE trained on a single graph(T-GAE single), the untrained T-GAE (T-GAE untrained), and GAE. The proposed training objective and encoder structure helps to prompt the expressiveness of GNN thus achieve higher accuracy as we introduce more perturbations.

(TGAE and WAlign) compared to traditional graph or node embedding techniques. In particular, `T-GAE` consistently achieves the best performance among all competing methods. This suggests that the encoder model illustrated in Fig. 1 and the training framework (7) illustrated in Fig. 2, provide an efficient approach to generate powerful node embedding, that is robust to real-world noise distributions for the task of network alignment, compared to the existing GNN frameworks.

5.4 Ablation study

The proposed architecture and training objective prompts the robustness of GNN when matching graphs with their highly perturbed versions. From Figure 4, we observe that T-GAE outperforms the untrained T-GAE by a great margin when matching highly permuted samples. This implies that the proposed training objective effectively improves the robustness of GNN, which is the key property to deploy GNN to match perturbed graphs. Further, the performance gap between GAE and T-GAE single underscores the efficacy of incorporating attention mechanism on each layer for both input and output node features, as illustrated in Figure 1.

6 Limitation

Although our approach achieves state-of-the-art performance in aligning real-graphs, on both graph matching and sub-graph matching tasks, approaching network alignment with a learning method, remains a heuristic and does not offer optimality guarantees. Furthermore, in order to process large graphs we cast network alignment as a self-supervised task. As a result in small-scale settings where the task can be tackled with computationally intensive efficient methods, our algorithm is not expected to perform the best. Finally, the complexity of T-GAE $\mathcal{O}(|\mathcal{V}|^2)$ is limiting, this bottleneck comes from the greedy linear assignment algorithm to match the node embedding, and therefore the alternative method with complexity $\mathcal{O}(|\mathcal{V}|c^2 + |\mathcal{E}|c + |\mathcal{V}|\text{log}(|\mathcal{V}|))$ should be deployed when we match very large scale graphs with their permuted versions.

7 Conclusion

We proposed T-GAE, a graph autoencoder framework that utilizes transferability and robustness of GNN to perform network alignment. T-GAE is an unsupervised approach that tackles the high computational cost of existing optimization based algorithms, and can be trained on multiple small to middle sized graphs to produce robust and permutation equivariant embeddings for larger scale unseen networks. We proved that the produced embeddings of GNNs are related to the spectral decomposition of the graph and are at least as good in graph matching as certain spectral methods. Our experiments with real-world benchmarks on both graph matching and sub-graph matching demonstrated the great potential of utilizing the good properties of GNNs to solve network optimization problems in a more efficient and scalable way.

References

Frank Emmert-Streib, Matthias Dehmer, and Yongtang Shi. Fifty years of graph matching, network alignment and network comparison. *Information sciences*, 346:180–197, 2016. 1

Shirin Nilizadeh, Apu Kapadia, and Yong-Yeol Ahn. Community-enhanced de-anonymization of online social networks. In *Proceedings of the 2014 acm sigsac conference on computer and communications security*, pages 537–548, 2014. 1

Kirk Ogaard, Heather Roy, Sue Kase, Rakesh Nagi, Kedar Sambhoos, and Moises Sudit. Discovering patterns in social networks with graph matching algorithms. In Ariel M. Greenberg, William G. Kennedy, and Nathan D. Bos, editors, *Social Computing, Behavioral-Cultural Modeling and Prediction*, pages 341–349, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-37210-0. 1

Rohit Singh, Jinbo Xu, and Bonnie Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences*, 105(35):12763–12768, 2008. 1, 7, 21

Jiazhou Chen, Hong Peng, Guoqiang Han, Hongmin Cai, and Jiulun Cai. HOGMMNC: a higher order graph matching with multiple network constraints model for gene–drug regulatory modules identification. *Bioinformatics*, 35(4):602–610, 07 2018a. ISSN 1367-4803. doi: 10.1093/bioinformatics/bty662. URL `https://doi.org/10.1093/bioinformatics/bty662`. 1

D. Conte, P. Foggia, C. Sansone, and M. Vento. Graph matching applications in pattern recognition and image processing. In *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, volume 2, pages II–21, 2003. doi: 10.1109/ICIP.2003.1246606. 1

Miklos Racz and Anirudh Sridhar. Correlated stochastic block models: Exact graph matching with applications to recovering communities. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 22259–22273. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/baf4f1a5938b8d520b328c13b51ccf11-Paper.pdf`. 1

Kurt M Anstreicher and Nathan W Brixius. Solving quadratic assignment problems using convex quadratic programming relaxations. *Optimization Methods and Software*, 16(1-4):49–68, 2001. 1

Joshua T Vogelstein, John M Conroy, Vince Lyzinski, Louis J Podrazik, Steven G Kratzer, Eric T Harley, Donniell E Fishkind, R Jacob Vogelstein, and Carey E Priebe. Fast approximate quadratic programming for graph matching. *PLOS one*, 10(4):e0121002, 2015. 1

Gunnar W Klau. A new graph-based method for pairwise global network alignment. *BMC bioinformatics*, 10(1):1–9, 2009. 1

Jiming Peng, Hans Mittelmann, and Xiaoxue Li. A new relaxation framework for quadratic assignment problems based on matrix splitting. *Mathematical Programming Computation*, 2:59–77, 2010. 1

Xingbo Du, Junchi Yan, and Hongyuan Zha. Joint link prediction and network alignment via cross-graph embedding. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 2251–2257. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/312. URL `https://doi.org/10.24963/ijcai.2019/312`. 1

Xingbo Du, Junchi Yan, Rui Zhang, and Hongyuan Zha. Cross-network skip-gram embedding for joint network alignment and link prediction. *IEEE Transactions on Knowledge and Data Engineering*, 34(3):1080–1095, 2022. doi: 10.1109/TKDE.2020.2997861. 1

Aritra Konar and Nicholas D Sidiropoulos. Graph matching via the lens of supermodularity. *IEEE Transactions on Knowledge and Data Engineering*, 34(5):2200–2211, 2020. 1

S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):695–703, 1988. doi: 10.1109/34.6778. 2, 4, 5, 7, 20

Soheil Feizi, Gerald Quon, Mariana Recamonde-Mendoza, Muriel Medard, Manolis Kellis, and Ali Jadbabaie. Spectral alignment of graphs. *IEEE Transactions on Network Science and Engineering*, 7(3):1182–1197, 2019. 2, 4

Si Zhang and Hanghang Tong. Final: Fast attributed network alignment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1345–1354, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939766. URL https://doi.org/10.1145/2939672.2939766. 2, 7, 19, 20, 21

Charilaos I Kanatsoulis and Nicholas D Sidiropoulos. Gage: Geometry preserving attributed graph embeddings. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 439–448, 2022. 2

Michele Berlingerio, Danai Koutra, Tina Eliassi-Rad, and Christos Faloutsos. Network similarity via multiple social theories. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 1439–1440, 2013. 2, 7, 20, 21

Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. Regal: Representation learning-based graph alignment. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 117–126, 2018. 2

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '14. ACM, August 2014. doi: 10.1145/2623330.2623732. URL http://dx.doi.org/10.1145/2623330.2623732. 2, 7, 20

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016a. 2

Xiyuan Chen, Mark Heimann, Fatemeh Vahedian, and Danai Koutra. Cone-align: Consistent network alignment with proximity-preserving node embedding. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1985–1988, 2020. 2, 7, 21

Paris A Karakasis, Aritra Konar, and Nicholas D Sidiropoulos. Joint graph embedding and alignment with spectral pivot. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 851–859, 2021. 2

Thomas N. Kipf and Max Welling. Variational graph auto-encoders, 2016. URL https://arxiv.org/abs/1611.07308. 2, 5, 7, 20

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5812–5823. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/3fe230348e9a12c13120749e3f9fa4cd-Paper.pdf. 2

P. Gainza, F. Sverrisson, F. Monti, E. Rodolà, D. Boscaini, M. M. Bronstein, and B. E. Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192, February 2020. 2

Alexey Strokach, David Becerra, Carles Corbi-Verge, Albert Perez-Riba, and Philip M. Kim. Fast and flexible protein design using deep graph neural networks. *Cell Systems*, 11(4):402–411.e4, October 2020. 2

Dejun Jiang, Zhenxing Wu, Chang Yu Hsieh, Guangyong Chen, Ben Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. *Journal of Cheminformatics*, 13(1):12, dec 2021. 2

Xinyue Hu, Zenan Sun, Yi Nian, Yifang Dang, Fang Li, Jingna Feng, Evan Yu, and Cui Tao. Explainable graph neural network for alzheimer's disease and related dementias risk prediction, 2023. 2

Chenxin Li, Xinyu Liu, Cheng Wang, Yifan Liu, Weihao Yu, Jing Shao, and Yixuan Yuan. Gtp-4o: Modality-prompted heterogeneous graph learning for omni-modal biomedical representation, 2024. URL https://arxiv.org/abs/2407.05540. 2

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017. 2

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 10:974–983, June 2018. 2

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey, 2020. URL https://arxiv.org/abs/2011.02260. 2

X. Liu, R. Wang, D. Sun, J. Li, C. Youn, Y. Lyu, J. Zhan, D. Wu, X. Xu, M. Liu, X. Lei, Z. Xu, Y. Zhang, Z. Li, Q. Yang, and T. Abdelzaher. Influence pathway discovery on social media. In *2023 IEEE 9th International Conference on Collaboration and Internet Computing (CIC)*, pages 105–109, Los Alamitos, CA, USA, nov 2023a. IEEE Computer Society. doi: 10.1109/CIC58953.2023.00023. URL https://doi.ieeecomputersociety.org/10.1109/CIC58953.2023.00023. 2

Qikai Yang, Panfeng Li, Zhicheng Ding, Wenjing Zhou, Yi Nian, and Xinhe Xu. A comparative study on enhancing prediction in social network advertisement through data augmentation. *arXiv preprint arXiv:2404.13812*, 2024. 2

Matthias Fey, Jan E. Lenssen, Christopher Morris, Jonathan Masci, and Nils M. Kriege. Deep graph matching consensus, 2020. URL https://arxiv.org/abs/2001.09621. 2

Tianshu Yu, Runzhong Wang, Junchi Yan, and Baoxin Li. Learning deep graph matching with channel-independent embedding and hungarian attention. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJgBd2NYPH. 2

Zhiyuan Liu, Yixin Cao, Fuli Feng, Xiang Wang, Jie Tang, Kenji Kawaguchi, and Tat-Seng Chua. Training free graph neural networks for graph matching, 2022. URL https://arxiv.org/abs/2201.05349. 2

Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey, 2020. URL https://arxiv.org/abs/1812.04202. 2

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications, 2021. URL https://arxiv.org/abs/1812.08434. 2

Zhehan Liang, Yu Rong, Chenxin Li, Yunlong Zhang, Yue Huang, Tingyang Xu, Xinghao Ding, and Junzhou Huang. Unsupervised large-scale social network alignment via cross network embedding. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 1008–1017, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384469. doi: 10.1145/3459637.3482310. URL https://doi.org/10.1145/3459637.3482310. 2

Ji Gao, Xiao Huang, and Jundong Li. Unsupervised graph alignment with wasserstein distance discriminator. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 426–435, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325. doi: 10.1145/3447548.3467332. URL https://doi.org/10.1145/3447548.3467332. 2, 7, 20

Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction, 2018b. URL https://arxiv.org/abs/1710.10568. 2

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling, 2018c. URL https://arxiv.org/abs/1801.10247. 2

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 257–266, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330925. URL https://doi.org/10.1145/3292500.3330925. 2

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method, 2020. URL https://arxiv.org/abs/1907.04931. 2

Tjalling C Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, pages 53–76, 1957. 3

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955a. 3

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=ryGs6iA5Km. 4, 21

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2018. 4

Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Stability properties of graph neural networks. *IEEE Transactions on Signal Processing*, 68:5680–5695, 2020. 4

Alejandro Parada-Mayorga, Zhiyang Wang, Fernando Gama, and Alejandro Ribeiro. Stability of aggregation graph neural networks. *IEEE Transactions on Signal and Information Processing over Networks*, 9:850–864, 2023. doi: 10.1109/TSIPN.2023.3341408. 4

Luana Ruiz, Zhiyang Wang, and Alejandro Ribeiro. Graphon and graph neural network stability. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5255–5259, 2021. doi: 10.1109/ICASSP39728.2021.9414838. 4

Luana Ruiz, Fernando Gama, and Alejandro Ribeiro. Gated graph recurrent neural networks. *IEEE Transactions on Signal Processing*, 68:6303–6318, 2020. 4, 6

Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. In *IJCAI*, 2021. 4

Charilaos I Kanatsoulis and Alejandro Ribeiro. Graph neural networks are more powerful than we think. *arXiv preprint arXiv:2205.09801*, 2022. 4

Willem H Haemers and Edward Spence. Enumeration of cospectral graphs. *European Journal of Combinatorics*, 25(2):199–211, 2004. 5

Xuezhi Wang, Haohan Wang, and Diyi Yang. Measure and improve robustness in nlp models: A survey, 2022. 6

Jiashu He. Performance analysis of facial recognition: A critical review through glass factor, 2021. 6

Chenxin Li, Xin Lin, Yijin Mao, Wei Lin, Qi Qi, Xinghao Ding, Yue Huang, Dong Liang, and Yizhou Yu. Domain generalization on medical imaging classification using episodic training with task augmentation. *Computers in Biology and Medicine*, 141:105144, 2022. ISSN 0010-4825. doi: https://doi.org/10.1016/j.compbiomed.2021.105144. URL https://www.sciencedirect.com/science/article/pii/S0010482521009380. 6

Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. It's who you know: graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 663–671, 2011. 6

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks, 2016b. 7, 21

Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18. ACM, July 2018. doi: 10.1145/3219819.3220025. URL http://dx.doi.org/10.1145/3219819.3220025. 7, 21

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15. International World Wide Web Conferences Steering Committee, May 2015. doi: 10.1145/2736277.2741093. URL http://dx.doi.org/10.1145/2736277.2741093. 7, 21

Hongteng Xu, Dixin Luo, and Lawrence Carin. Scalable gromov-wasserstein learning for graph partitioning and matching, 2019b. 7, 21

Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. Tri-party deep network representation. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1895–1901. IJCAI/AAAI Press, 2016. 8, 19

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop, NeurIPS 2018*, 2018. 8, 19, 24

Jiashu He, Mingyu Derek Ma, Jinxuan Fan, Dan Roth, Wei Wang, and Alejandro Ribeiro. Give: Structured reasoning with knowledge graph inspired veracity extrapolation, 2024. URL https://arxiv.org/abs/2410.08475. 15

Congcong Ge, Xiaoze Liu, Lu Chen, Baihua Zheng, and Yunjun Gao. Make it easy: An effective end-to-end entity alignment framework. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 777–786, New York, NY, USA, 2021a. Association for Computing Machinery. ISBN 9781450380379. doi: 10.1145/3404835.3462870. URL https://doi.org/10.1145/3404835.3462870. 15

Congcong Ge, Xiaoze Liu, Lu Chen, Yunjun Gao, and Baihua Zheng. Largeea: aligning entities for large-scale knowledge graphs. *Proceedings of the VLDB Endowment*, 15(2):237–245, October 2021b. ISSN 2150-8097. doi: 10.14778/3489496.3489504. URL http://dx.doi.org/10.14778/3489496.3489504. 15

Yunjun Gao, Xiaoze Liu, Junyang Wu, Tianyi Li, Pengfei Wang, and Lu Chen. Clusterea: Scalable entity alignment with stochastic training and normalized mini-batch similarities. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 421–431, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539331. URL https://doi.org/10.1145/3534678.3539331. 15

Congcong Ge, Pengfei Wang, Lu Chen, Xiaoze Liu, Baihua Zheng, and Yunjun Gao. Collaborem: A self-supervised entity matching framework using multi-features collaboration. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12139–12152, 2023. doi: 10.1109/TKDE.2021.3134806. 15

Xiaoze Liu, Junyang Wu, Tianyi Li, Lu Chen, and Yunjun Gao. Unsupervised entity alignment for temporal knowledge graphs. In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 2528–2538, New York, NY, USA, 2023b. Association for Computing Machinery. ISBN 9781450394161. doi: 10.1145/3543507.3583381. URL https://doi.org/10.1145/3543507.3583381. 15

Bolin Zhu, Xiaoze Liu, Xin Mao, Zhuo Chen, Lingbing Guo, Tao Gui, and Qi Zhang. Universal multi-modal entity alignment via iteratively fusing modality similarity paths, 2023. URL https://arxiv.org/abs/2310.05364. 15

Jian Ding, Zongming Ma, Yihong Wu, and Jiaming Xu. Efficient random graph matching via degree profiles, 2020. 15

Yihong Wu, Jiaming Xu, and Sophie H. Yu. Settling the sharp reconstruction thresholds of random graph matching, 2022. 15

Cheng Mao, Yihong Wu, Jiaming Xu, and Sophie H. Yu. Random graph matching at otter's threshold via counting chandeliers, 2023. 15

Stefania Sardellitti, Sergio Barbarossa, and Paolo Di Lorenzo. On the graph fourier transform for directed graphs. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):796–811, 2017. 16

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955b. 18, 24

Jérôme Kunegis. Konect: the koblenz network collection. *Proceedings of the 22nd International Conference on World Wide Web*, 2013. 19, 24

Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014. 19

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008. 19

Si Zhang and Hanghang Tong. Attributed network alignment: Problem definitions and fast solutions. *IEEE Transactions on Knowledge and Data Engineering*, 31:1680–1692, 2019. URL https://api.semanticscholar.org/CorpusID:70142000. 19

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. 20

A Appendix

B Notation

Our notation is summarized in Table 1.

Table 1: Key notations used in this paper.

$\mathcal{G}$	$\triangleq$	Graph
$\mathcal{V}$	$\triangleq$	Set of nodes
$\mathcal{E}$	$\triangleq$	Set of edges
N	$\triangleq$	Number of nodes
$\boldsymbol{D}$	$\triangleq$	Degree matrix
$\boldsymbol{S}$	$\triangleq$	$\{0,1\}^{N\times N}$ adjacency matrix
$\boldsymbol{X}$	$\triangleq$	$N \times D$ feature matrix
$\boldsymbol{H}$	$\triangleq$	aggregation results of GNN convolution
$\boldsymbol{W}$	$\triangleq$	weight matrix of the Graph Neural Network
$\boldsymbol{N}_v$	$\triangleq$	neighbors of node v
$\boldsymbol{I}$	$\triangleq$	Identity matrix
$\mathbf{0}$	$\triangleq$	vector or matrix of zeros
$\boldsymbol{A}^T$	$\triangleq$	transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{rc}$	$\triangleq$	entry at r-th row and j-th column of matrix $\boldsymbol{A}$
$\|\cdot\|_F$	$\triangleq$	Frobenius norm

C Network Alignment in broader domains

In this paper, we study the general form of network alignment to match nodes between two graphs. This task is challenging because of the little given information. However, it's not trivial to correctly and efficiently utilize the information contained in text-rich networks, such as KGs [He et al., 2024]. In the domain of knowledge graph (KG) mining, entity alignment (EA) is the task to identify the same entities existing in knowledge graphs. [Ge et al., 2021a] proposes to replace the labor-intensive pre-processing with entity names mining. A structural-based refinement procedure is then applied to refine the entity name matching results. [Ge et al., 2021b, Gao et al., 2022] solves large scale EA by aligning the KG in mini-batches. [Ge et al., 2023] further proposes a self-supervised EA framework by automatically generating positive and negative matched node pairs. [Liu et al., 2023b] generalizes the unsupervised matching algorithm to temporal KGs. Specifically, it encodes the temporal and relational information respectively before an innovative jointly decoding process. Recently, the ability to deploy EA algorithms in real-world scenarios is enhanced by [Zhu et al., 2023], which aligns multi-modal knowledge graphs.

Network alignment, as an important problem, has been studied not only by the community of data mining, it has also been mathematically and statistically investigated. A number of approaches have been proposed to solve this problem for Erdős Rényi random graphs $G(n, \frac{d}{n})$. It has been proved that a perfectly true vertex correspondence can be recovered in polynomial time with high probability [Ding et al., 2020]. Furthermore, a sharp threshold has been proved for both Erdős Rényi model and Gaussian model [Wu et al., 2022]. Most recently, a novel approach to calculate similarity scores based on counting weighted trees rooted at each vertex has been proposed Mao et al. [2023]. Such approach has been proved to be effective in solving the aforementioned network alignment problem on random graphs with high probability. Readers are encouraged to refer to the authors of these publications [Ding et al., 2020, Wu et al., 2022, Mao et al., 2023] for further reading.

D Proof of Theorem 3.2

D.1 Spectral characterization of GNNs

What remains to be answered is the ability of a GNN to approximate a function that performs graph alignment. To understand the function approximation properties of GNNs we study them in the spectral domain. To this end, we consider the recursive formula in (4) where f is the summation function and g is multivariate linear for $K-2$ layers, and the MLP in the $(K-1)$-th layer. The overall operation can be written in a matrix form as:

$$\boldsymbol{X}^{(l+1)} = \sigma\left(\sum_{k=0}^{K-1} \boldsymbol{S}^k \boldsymbol{X}^{(l)} \boldsymbol{H}_k^{(l)}\right), \tag{8}$$

where $\boldsymbol{H}_k^{(l)} \in \mathbb{R}^{D^{l+1} \times D^l}$ is a linear mapping. Computing the spectral decomposition of $\boldsymbol{S}$ yields:

$$\boldsymbol{X}^{(l+1)} = \sigma\left(\sum_{k=0}^{K-1} \boldsymbol{V}\boldsymbol{\Lambda}^k \boldsymbol{V}^T \boldsymbol{X}^{(l)} \boldsymbol{H}_k^{(l)}\right) = \sigma\left(\sum_{k=0}^{K-1}\sum_{n=1}^{N} \lambda_n^k \boldsymbol{v}_n \boldsymbol{v}_n^T \boldsymbol{X}^{(l)} \boldsymbol{H}_k^{(l)}\right). \tag{9}$$

Then each each column of $\boldsymbol{X}^{(l+1)}$ can be written as

$$\boldsymbol{X}^{(l+1)}[:,i] = \sigma\left(\sum_{k=0}^{K-1}\sum_{n=1}^{N} \lambda_n^k \boldsymbol{v}_n \boldsymbol{v}_n^T \boldsymbol{X}^{(l)} \boldsymbol{H}_k^{(l)}[:,i]\right) = \sigma\left(\sum_{n=1}^{N} a_n^{(i)} \boldsymbol{v}_n\right), \tag{10}$$

where $\lambda_n, \boldsymbol{v}_n$ are the $n-$th eigenvalue and eigenvector and $a_n^{(i)} = \boldsymbol{v}_n^T \boldsymbol{X}^{(l)} \sum_{k=0}^{K-1} \lambda_n^k \boldsymbol{H}_k^{(l)}[:,i]$ is a scalar related to the Graph Fourier Transform (GFT) of $\boldsymbol{X}^{(l)}$ [Sardellitti et al., 2017]. It is clear from equation (10) that the output of each layer is a linear combination of the adjacency eigenvectors, followed by a pointwise non-linearity. Thus, a GNN can produce unique and more powerful graph embeddings than spectral methods by processing the eigenvectors and eigenvalues of the adjacency matrix. To prove Theorem 3.2. We consider one layer GNN with a vector input $\boldsymbol{x} \in \mathbb{R}^N$. This GNN can be represented by the following equation:

$$\boldsymbol{Y} = \sigma\left(\sum_{k=0}^{K-1} \boldsymbol{S}^k \boldsymbol{x} \boldsymbol{h}_k^T\right), \tag{11}$$

where $\boldsymbol{h}_k \in \mathbb{R}^m$ and $\boldsymbol{x}\boldsymbol{h}_k^T$ is an outer-product operation. The equation in (11) describes a set of m graph filters of the form:

$$\boldsymbol{y}_i = \sigma\left(\sum_{k=0}^{K-1} h_k^i \boldsymbol{S}^k \boldsymbol{x}\right), \quad \text{for } i = 1, \ldots, m \tag{12}$$

D.2 White random input and variance computation

Let $\boldsymbol{x}$ be a white random vector with $\mathbb{E}[\boldsymbol{x}] = 0$ and $\mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^T\right] = \boldsymbol{I}$, where $\boldsymbol{I}$ is the diagonal matrix. Also let $\sigma(\cdot) = (\cdot)^2$ be the elementwise square function. Then (12) can be written as:

$$\boldsymbol{y}_i = \left(\sum_{k=0}^{K-1} h_k^i \boldsymbol{S}^k \boldsymbol{x}\right)^2 = \text{diag}\left(\sum_{k=0}^{K-1} h_k^i \boldsymbol{S}^k \boldsymbol{x}\boldsymbol{x}^T \sum_{j=0}^{K-1} h_j^i \boldsymbol{S}^{j^T}\right) \tag{13}$$

Since $\boldsymbol{x}$ is a random vector $\boldsymbol{y}_i$ is also a random vector. The expected value of $\boldsymbol{y}_i$ yields:

$$\mathbb{E}[\boldsymbol{y}_i] = \mathbb{E}\left[\text{diag}\left(\sum_{k=0}^{K-1} h_k^i \boldsymbol{S}^k \boldsymbol{x}\boldsymbol{x}^T \sum_{j=0}^{K-1} h_j^i \boldsymbol{S}^{j^T}\right)\right] \tag{14}$$

$$= \text{diag}\left(\sum_{k=0}^{K-1} h_k^i \boldsymbol{S}^k \mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^T\right] \sum_{j=0}^{K-1} h_j^i \boldsymbol{S}^{j^T}\right)$$

$$= \text{diag}\left(\sum_{k=0}^{K-1} h_k^i \boldsymbol{S}^k \sum_{j=0}^{K-1} h_j^i \boldsymbol{S}^{j^T}\right) \tag{15}$$

D.3 Single band filtering

In the second part of the proof we study the graph filter using the spectral decomposition of the graph:

$$\boldsymbol{y} = \sum_{k=0}^{K-1} h_k \boldsymbol{S}^k \boldsymbol{x} \tag{16}$$
$$= \sum_{k=0}^{K-1} h_k \boldsymbol{V}\boldsymbol{\Lambda}^k \boldsymbol{V}^T \boldsymbol{x} \tag{17}$$
$$= \sum_{k=0}^{K-1} h_k \sum_{n=1}^{N} \lambda_n^k \boldsymbol{v}_n \boldsymbol{v}_n^T \boldsymbol{x} \tag{18}$$
$$= \sum_{n=1}^{N} \boldsymbol{v}_n^T \boldsymbol{x} \sum_{k=0}^{K-1} h_k \lambda_n^k \boldsymbol{v}_n. \tag{19}$$

Let us focus on the following polynomial:

$$\tilde{h}\left(\lambda\right) = \sum_{k=0}^{K-1} h_k \lambda^k, \tag{20}$$

that represents a graph filter in the frequency domain by. For q distinct eigenvalues we can write a system of linear equations using the polynomial in (20):

$$\begin{bmatrix} \tilde{h}\left(\lambda_1\right) \\ \tilde{h}\left(\lambda_2\right) \\ \vdots \\ \tilde{h}\left(\lambda_q\right) \end{bmatrix} = \begin{bmatrix} 1 \ \lambda_1 \ \lambda_1^2 \dots \lambda_1^{K-1} \\ 1 \ \lambda_2 \ \lambda_2^2 \dots \lambda_2^{K-1} \\ \vdots \\ 1 \ \lambda_q \ \lambda_q^2 \dots \lambda_q^{K-1} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{K-1} \end{bmatrix} = \boldsymbol{W}\boldsymbol{h} \tag{21}$$

$\boldsymbol{W}$ is a Vandermonde matrix and when $K = q$ the determinant of $\boldsymbol{W}$ takes the form:

$$\det\left(\boldsymbol{W}\right) = \prod_{1 \le i < j \le q} \left(\lambda_i - \lambda_j\right) \tag{22}$$

Since the values λ_i are distinct, $\boldsymbol{W}$ has full column rank and there exists a graph filter with unique parameters $\boldsymbol{h}$ that passes only the λ eigenvalue, i.e.,

$$\tilde{h}\left(\lambda_i\right) = \begin{cases} 1, & \text{if } \ \lambda_i = \lambda \\ 0, & \text{if } \ \lambda_i \neq \lambda \end{cases} \tag{23}$$

Under this parametrization, equation (16) takes the form $\boldsymbol{y} = \boldsymbol{v}_\lambda \boldsymbol{v}_\lambda^T \boldsymbol{x}$, where $\boldsymbol{v}_\lambda$ is the eigenvector corresponding to λ.

D.4 GNN and absolute eigenvectors

Using the previous analysis we can design parameters h_k such that:

$$\sum_{k=0}^{K-1} h_k \boldsymbol{S}^k = \boldsymbol{v}_\lambda \boldsymbol{v}_\lambda^T \tag{24}$$

and then equation (14) takes the form:

$$\mathbb{E}\left[\boldsymbol{y}_i\right] = \text{diag}\left(\sum_{k=0}^{K-1} h_k^i \boldsymbol{S}^k \sum_{j=0}^{K-1} h_j^i \boldsymbol{S}^{j^T}\right) \tag{25}$$
$$= \text{diag}\left(\boldsymbol{v}_\lambda \boldsymbol{v}_\lambda^T \boldsymbol{v}_\lambda \boldsymbol{v}_\lambda^T\right) \tag{26}$$
$$= \text{diag}\left(\boldsymbol{v}_\lambda \boldsymbol{v}_\lambda^T\right) \tag{27}$$
$$= \left|\boldsymbol{v}_\lambda\right|^2 \tag{28}$$

We can therefore design $\boldsymbol{h}_k \in \mathbb{R}^m$ for $k = 0, \ldots, m-1$ to compute the absolute value of m eigenvectors of $\boldsymbol{S}$ that correspond to the top m distinct eigenvalues, i.e.,

$$\mathbb{E}\left[\boldsymbol{y}_i\right] = |\boldsymbol{u}_i|^2, \quad i = 1, \ldots, m \tag{29}$$
$$\tag{30}$$

We can do the same for graph $\hat{\boldsymbol{S}}$ and compute:

$$\mathbb{E}\left[\hat{\boldsymbol{y}}_i\right] = |\hat{\boldsymbol{u}}_i|^2, \quad i = 1, \ldots, m \tag{31}$$
$$\tag{32}$$

Since both $\boldsymbol{S},\ \hat{\boldsymbol{S}}$ have distinct eigenvalues, we can concatenate the output of each neuron and result in layer-1 outputs as:

$$\boldsymbol{Y}^{(1)} = |\boldsymbol{U}|, \quad \hat{\boldsymbol{Y}}^{(1)} = |\hat{\boldsymbol{U}}| \tag{33}$$

As a result, the previously described GNN can a least yield the same alignment accuracy as the absolute values of the eigenvectors.

D.5 Generalization to multiple graph pairs

The analysis in the previoius subsections is indeed presented for a pair of graphs but can be directly extended for any set of graphs. We can generalize the Theorem 3.2, to read as: Let $\{\mathcal{G}_1, \ldots, \mathcal{G}_M\}$ be a set of graphs with adjacencies $\{\boldsymbol{S}_1, \ldots, \boldsymbol{S}_M\}$ that have non-repeated eigenvalues. Then for any $\boldsymbol{S},\ \hat{\boldsymbol{S}} \in \{\boldsymbol{S}_1, \ldots, \boldsymbol{S}_M\}$, there exists a GNN $\phi\left(\boldsymbol{X}; \boldsymbol{S}, \mathcal{H}\right) : \mathbb{R}^{N \times D} \to \mathbb{R}^{N \times D^L}$ such that:

$$\left\| \boldsymbol{S} - \boldsymbol{P}^{\diamond} \hat{\boldsymbol{S}} \boldsymbol{P}^{\diamond^T} \right\|_F^2 \leq \left\| \boldsymbol{S} - \boldsymbol{P}^{*} \hat{\boldsymbol{S}} \boldsymbol{P}^{*^T} \right\|_F^2 \leq \left\| \boldsymbol{S} - \check{\boldsymbol{P}} \hat{\boldsymbol{S}} \check{\boldsymbol{P}}^{T} \right\|_F^2$$

with

$$\boldsymbol{P}^* = \underset{\boldsymbol{P} \in \mathcal{P}}{\arg\min} \left\| \phi\left(\boldsymbol{X}; \boldsymbol{S}, \mathcal{H}\right) - \boldsymbol{P} \phi\left(\hat{\boldsymbol{X}}; \hat{\boldsymbol{S}}, \mathcal{H}\right) \right\|_F^2,$$

where $\boldsymbol{P}^{\diamond},\ \check{\boldsymbol{P}}$ are solutions to the optimization problems in (1) and (3) respectively.

E Implementation Details

In this section we discuss the implementation details of our framework.

E.1 Assignment Optimization

The proposed `T-GAE` learns learns a GNN encoder that can produce node representations for different graphs. Let $\phi\left(\boldsymbol{X}; \boldsymbol{S}, \mathcal{H}\right)$ represent the embeddings of the nodes corresponding to the graph with adjacency $\boldsymbol{S}$ and $\phi\left(\hat{\boldsymbol{X}}; \hat{\boldsymbol{S}}, \mathcal{H}\right)$ represent the embeddings of the nodes corresponding to the graph with adjacency $\hat{\boldsymbol{S}}$. Then network alignment boils down to solving the following optimization problem:

$$\min_{\boldsymbol{P} \in \mathcal{P}} \left\| \phi\left(\boldsymbol{X}; \boldsymbol{S}, \mathcal{H}\right) - \boldsymbol{P} \phi\left(\hat{\boldsymbol{X}}; \hat{\boldsymbol{S}}, \mathcal{H}\right) \right\|_F^2. \tag{34}$$

The problem in (34) can be optimally solved in $\mathcal{O}\left(N^3\right)$ flops by the Hungarian algorithm [Kuhn, 1955b]. To avoid this computational burden we employ the greedy Hungarian approach that has computational complexity $\mathcal{O}\left(N^2\right)$ and usually works well in practice.

The greedy Hungarian approach is described in Algorithm 1. For each row of $\phi\left(\boldsymbol{X}; \boldsymbol{S}, \mathcal{H}\right),\ \phi\left(\hat{\boldsymbol{X}}; \hat{\boldsymbol{S}}, \mathcal{H}\right)$, which corresponds to the node embeddings of the different graphs, we compute the pairwise Euclidean distance which is stores in the distance matrix $\boldsymbol{D}$. Then, at each iteration, we find the nodes with the smallest distance and remove the aligned pairs from $\boldsymbol{D}$. This process is repeated until all the nodes are paired up for alignment.

Algorithm 1: Greedy Hungarian Algorithm

```
Input: Feature matrices $\boldsymbol{X}$, $\hat{\boldsymbol{X}}$
Output: Assignment Matrix
1  $\boldsymbol{P} := \mathbf{0}_{N\times N}$                                   // Initialize permutation matrix
2  $\boldsymbol{D} :=$ PairwiseDistance $\left(\boldsymbol{X}, \hat{\boldsymbol{X}}\right)$        // pairwise Euclidean distance
3  rows := 0,1,...,$N$-1                                      // Corresponds to $\boldsymbol{X}$
4  cols := 0,1,...,$N$-1                                      // Corresponds to $\hat{\boldsymbol{X}}$
   /* Iterate to assign node pairs with minimum Euclidean distance    */
5  for $n$=1 to $N$ do
6      $i, j :=$ argmin $(\boldsymbol{D})$
7      $r :=$ rows $[i]$
8      $c :=$ cols $[j]$
9      $\boldsymbol{P}_{rc} := 1$
10     Remove $r$ from rows
11     Remove $c$ from cols
12     Remove the $i$-th row from $\boldsymbol{D}$
13     Remove the $j$-th column from $\boldsymbol{D}$
14 return $\boldsymbol{P}$
```

Table 2: Summary of Dataset statistics that are included in Section 5

Task	Dataset	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	# Aligned Edges	Network Type
Graph Matching	Celegans [Kunegis, 2013]	453	2,025	2,025	Interactome
	Arenas [Leskovec and Krevl, 2014]	1,133	5,451	5,451	Email Communication
	Cora [Sen et al., 2008]	2,708	5,278	5,278	Citation Network
	Douban [Zhang and Tong, 2016]	3,906	7,215	7,215	Social Network
	Dblp [Pan et al., 2016]	17,716	52,867	52,867	Citation Network
	Coauthor CS [Shchur et al., 2018]	18,333	81,894	81,894	Coauthor Network
Subraph Matching	ACM-DBLP [Zhang and Tong, 2019]	9,872 9,916	39,561 44,808	6,352	Citation Network
	Douban Online-Offline [Zhang and Tong, 2016]	3,906 1,118	1,632 3,022	1,118	Social Network

E.2 Datasets

We include statistics of the datasets used in our experiments in Table 2. The detailed descriptions of each dataset are presented below:

- Celegans [Kunegis, 2013]: The vertices represent proteins and the edges their protein-protein interactions.
- Arenas Email [Leskovec and Krevl, 2014]: The email communication network at the University Rovira i Virgili in Tarragona in the south of Catalonia in Spain. Nodes are users and each edge represents that at least one email was sent.
- Douban [Zhang and Tong, 2016]: Contains user-user relationship on the Chinese movie review platform. Each edge implies that two users are contacts or friends.
- Cora [Sen et al., 2008]: The dataset consists of 2708 scientific publications, with edges representing citation relationships between them. Cora has been one of the major benchmark datasets in many graph mining tasks.
- Dblp [Pan et al., 2016]: A citation network dataset that is extracted from DBLP, Association for Computing Machinery (ACM), Microsoft Academic Graph (MAG), and other sources. It is considered a benchmark in multiple tasks.
- Coauthor_CS [Shchur et al., 2018]: The coauthorship graph is generated from MAG. Nodes are the authors and they are connected with an edge if they coauthored at least one paper.
- ACM$-$DBLP [Zhang and Tong, 2019]: The citation networks that share some common nodes. The task is to identify the publications that appear in both networks.

- Douban Online−Offline [Zhang and Tong, 2016]: The two social networks contained in this dataset represents the online and offline events of the Douban social network. The task is to identify users that participate in both online and offline events.

E.3 Baselines

E.3.1 Graph Neural Network(GNN) based methods

To have a fair comparison with the node embedding models, GAE is implemented using the same set of parameters as T-GAE, which can be found at Section E.4. WAlign is implemented using parameters suggested by the author. We report the best results they achieved during training.

- `WAlign` [Gao et al., 2021] fits a GNN to each of the input graphs, trains the model by reconstructing the given inputs and minimizing an approximation of Wasserstein distance between the node embeddings. We use the author's implementation from `https://github.com/gaoji7777/walign.git`.
- `GAE`, `VGAE`[Kipf and Welling, 2016] are self-supervised graph learning frameworks that are trained by reconstructing the graph. The encoder is a GCN[Kipf and Welling, 2017] and linear decoder is applied to predict the original adjacency. In VGAE, Gausian Noise is introduced to the node embeddings before passing to the decoder. We use the implementation from `https://github.com/DaehanKim/vgae_pytorch`. We train GAE by reconstructing the given network using the netsimile node embedding.

E.3.2 Graph/Node embedding techniques

- `NetSimile` [Berlingerio et al., 2013] uses the structural features described earlier to match the nodes of the graphs. Since the `NetSimile` features are used as input to the `T-GAE`, they provide a measure to assess the benefit of using `T-GAE` for node embedding. It proposed 7 egonet-based features, to measure network similarity. We process these features by Algorithm 1 to perform network alignment. The 7-dimensional Netsimile features are:
  - d_i = degree of node i
  - c_i = number of triangles connected to node i over the number of connected triples centered on node i
  - $\bar{d}_{N_i} = \frac{1}{d_i} \sum_{j \in N_i} d_j$, average number of two-hop neighbors
  - $\bar{c}_{N_i} = \frac{1}{d_i} \sum_{j \in N_i} c_j$, average clustering coefficient
  - Number of edges in node i's egonet
  - Number of outgoing edges from node i's egonet
  - Number of neighbors in node i's egonet

  The implementation is based on netrd library where we use the feature extraction function. The source code can be found at `https://netrd.readthedocs.io/en/latest/_modules/netrd/distance/netsimile.html`
- `Spectral` [Umeyama, 1988] It solves the following optimization problem:

$$\min_{\boldsymbol{P} \in \mathcal{P}} \left\| \, |\boldsymbol{V}| - \boldsymbol{P} \left| \hat{\boldsymbol{V}} \right| \, \right\|_F^2 , \tag{35}$$

  where $\boldsymbol{V}$, $\hat{\boldsymbol{V}}$ are the eigenvectors corresponding to the adjacencies of the graphs that we want to match. In our initial experiments, we observed that a subset of the eigenvectors yields improved results compared to the whole set. We tried $1 - 10$ top eigenvectors and concluded that 4 eigenvectors are those that yield the best results on average. Thus we solve the above problem with the top-4 eigenvectors.
- `DeepWalk` [Perozzi et al., 2014]: A node embedding approach, simulates random walks on the graph and apply skip-gram on the walks to generate node embedding. We use the implementation from Karateclub. The algorithm is implemented with the default parameters as suggested by this repository, the number of random walks is 10 with each walk of length 80. The dimensionality of embedding is set to be 128. We run the algorithm with 1 epoch and set the learning rate to be 0.05.

- `Node2Vec` [Grover and Leskovec, 2016b]: An improve version of DeepWalk, it has weights on the randomly generated random walks, to make the neighborhood preserving objective more flexible. We use the implementation from Karateclub. The default parameters are used. We simulate 10 random walks on the graph with length 80. p and q are both equal to 1. Dimensionality of embeddings is set to be 4 and we run 1 epoch with learning rate 0.05.
- `GraphWave` [Donnat et al., 2018]: The structure information of the graphs is captured by simulating heat diffusion process on them. We use the implementation from Karateclub with the default parameters: number of evaluation points is 200, step size is 0.1, heat coefficient is 1.0 and Chebyshev polynomial order is set to be 100. Note that this implementation does not work on graphs with more than 10,000 nodes, so we exclude this model on the DBLP and Coauthor_CS dataset.
- `LINE` [Tang et al., 2015]: An optimization based graph embedding approach that aims to preserve local and global structures of the network by considering substructures and structural-quivariant nodes. We use the PyTorch implementation from `https://github.com/zxhhh97/ABot`. All parameters are set to default as the authors suggested.

E.3.3 Optimization based graph matching algorithms

- `FINAL` [Zhang and Tong, 2016] is an optimization approach, following an alignment consistency principle, and tries to match nodes with similar topology. In the case of graphs without attributes `FINAL` is equivalent to the popular
- `Isorank` [Singh et al., 2008] algorithm, whereas using `NetSimile` as an input to `FINAL` resulted in inferior performance and was therefore omitted. We use the code in `https://github.com/sizhang92/FINAL-KDD16` with $\boldsymbol{H}$ being the degree similarity matrix, $\alpha = 0.8$, $\text{maxiter} = 30$, $\text{tol} = 1e-4$ as suggested in the repository.
- `ConeAlign` [Chen et al., 2020] is a graph embedding based approach. The matching is optimized in each iteration by the Wasserstein Procrustes distances between the matched embeddings calculated on a mini batch in order to preserve scalability. We use the official implementation from `https://github.com/GemsLab/CONE-Align` and preserved all the suggested parameters.
- `S-GWL` [Xu et al., 2019b] matches two given graphs by retrieving node correspondence from the optimal transport associated with the Gromov-Wasserstein discrepancy between the graphs. We use the implementation by the authors in `https://github.com/HongtengXu/s-gwl`, since the performance of S-GWL is very sensitive to the parameter gamma, as suggested by the authors, we fine-tuned this parameter over the range of $[0.001, 0.1]$ for each dataset on the cleaned graph, and use that optimal parameter for all other experiments on this dataset.

E.4 T-GAE model details

As illustrated in Figure 1, the structure of our proposed encoder consists of two MLPs and a series of GNN layers. The node features are processed by a 2-layer MLP and passed to all the GNN layers. We add skip connections between this MLP layer and all the subsequent GNN layers. The outputs of all GNN layers are concatenated and passed to another 2-layer MLP, followed by a linear decoder to generate the reconstructed graph. The model is optimized end to end by equation 7. For graph matching experiments, since we consider the general case where graphs are given without node attributes, we use the 7 structural features proposed in [Berlingerio et al., 2013]. The features include the degree of each node, the local and average clustering coefficient, and the number of edges, outgoing edges, and neighbors in each node's egonet. This input feature is applied for all GNN-based methods. As a result, the performance of `NetSimile`, vanilla `GAE` and `WAlign` provide measures to assess the benefit of using `T-GAE` for node embedding. Note that one can choose different message passing functions as f and g in Equation (4), and any structure-preserving node features. Our reported results are based on `GIN` [Xu et al., 2019a] and `Netsimile` [Berlingerio et al., 2013].

F More Baseline Results

We present the graph matching accuracy for the baseline methods that are not permutation equavariant in Table 3, and sub-graph matching accuracy on Douban Online-Offline dataset for GraphWave in Table 4, as it is not scalable on ACM/DBLP.

Table 3: Graph matching accuracy on 10 randomly perturbed samples under different levels of edge editing for VGAE, LINE and DeepWalk.

	Dataset	VGAE	LINE	DeepWalk
0%	Celegans	0.3 ± 0.1	1.0 ± 0.5	1.8 ± 0.6
	Arenas	0.1 ± 0.1	0.2 ± 0.1	0.3 ± 0.2
	Douban	0.0 ± 0.0	0.0 ± 0.0	0.1 ± 0.0
	Cora	0.1 ± 0.0	0.0 ± 0.0	0.1 ± 0.0
1%	Celegans	0.3 ± 0.1	1.0 ± 0.4	1.2 ± 0.5
	Arenas	0.1 ± 0.1	0.1 ± 0.1	0.3 ± 0.1
	Douban	0.0 ± 0.0	0.0 ± 0.0	0.1 ± 0.0
	Cora	0.1 ± 0.1	0.1 ± 0.0	0.2 ± 0.1
5%	Celegans	0.6 ± 0.3	0.9 ± 0.3	1.0 ± 0.3
	Arenas	0.2 ± 0.1	0.2 ± 0.2	0.2 ± 0.1
	Douban	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	Cora	0.1 ± 0.0	0.1 ± 0.0	0.1 ± 0.0

Table 4: Sub-graph matching performance for GraphWave on Douban Online-Offline

Hit rate	GraphWave
Hit@1	0.09
Hit@5	0.36
Hit@10	0.81
Hit@50	4.74
Hit@100	9.12

G Degree Perturbation Model Results

Table 5: Graph matching accuracy on 10 randomly perturbed samples under different levels of edge removal on Degree model. The proposed T-GAE is trained on the clean Celegans, Arena, Douban, and Cora networks, and tested on noisy versions of them and the larger Dblp, and Coauthor CS. Accuracy above 80% is highlighted in green, 60% to 80% accuracy is in yellow, and performance below 60% is in red.

	Dataset \ Algorithm	***Feature Engineering based***			***Optimization based***			***GNN based***		
		Spectral	Netsimile	GraphWave	FINAL	S-GWL	ConeAlign	WAlign	GAE	**T-GAE**
0% perturbation	Celegans	87.8 ± 1.5	72.7 ± 0.9	65.3 ± 1.7	92.2 ± 1.2	$\mathbf{93.0 \pm 1.5}$	66.6 ± 1.2	88.4 ± 1.6	90.9 ± 2.6	91.0 ± 1.1
	Arenas	97.7 ± 0.4	94.7 ± 0.3	81.7 ± 0.7	97.5 ± 0.3	97.5 ± 0.3	87.8 ± 0.6	97.4 ± 0.5	97.6 ± 0.4	$\mathbf{97.8 \pm 0.4}$
	Douban	89.9 ± 0.4	46.4 ± 0.4	17.5 ± 0.2	89.9 ± 0.3	$\mathbf{90.1 \pm 0.3}$	68.1 ± 0.4	90.0 ± 0.4	89.5 ± 0.4	$\mathbf{90.1 \pm 0.3}$
	Cora	85.0 ± 0.4	73.7 ± 0.4	8.3 ± 0.4	87.5 ± 0.7	87.3 ± 0.7	38.5 ± 0.7	87.2 ± 0.4	87.1 ± 0.8	$\mathbf{87.5 \pm 0.4}$
	Dblp	84.5 ± 0.1	63.7 ± 0.2	doesn't scale	$\mathbf{85.6 \pm 0.2}$	> 48 hours	44.3 ± 0.6	85.6 ± 0.2	85.2 ± 0.3	$\mathbf{85.6 \pm 0.2}$
	Coauthor CS	97.5 ± 0.1	90.9 ± 0.1	doesn't scale	$\mathbf{97.6 \pm 0.1}$	> 48 hours	75.8 ± 0.5	97.5 ± 0.2	97.6 ± 0.3	$\mathbf{97.6 \pm 0.1}$
1% perturbation	Celegans	21.8 ± 16.9	63.1 ± 1.4	4.9 ± 1.1	62.5 ± 4.1	$\mathbf{73.1 \pm 11.7}$	61.4 ± 3.2	70.7 ± 4.4	7.8 ± 2.5	63.4 ± 6.1
	Arenas	66.0 ± 21.6	90.9 ± 0.7	5.8 ± 2.6	56.7 ± 2.7	92.3 ± 1.2	86.1 ± 0.6	96.0 ± 0.5	0.9 ± 0.5	$\mathbf{96.7 \pm 0.6}$
	Douban	9.8 ± 13.5	37.6 ± 0.6	0.6 ± 0.3	35.0 ± 1.1	69.9 ± 1.2	59.6 ± 2.2	81.3 ± 1.9	0.4 ± 0.0	$\mathbf{83.7 \pm 2.2}$
	Cora	25.3 ± 13.4	64.4 ± 1.2	0.1 ± 0.0	32.0 ± 2.0	31.1 ± 4.4	30.6 ± 2.7	74.6 ± 0.7	28.2 ± 0.3	$\mathbf{81.0 \pm 2.5}$
	Dblp	3.4 ± 0.8	52.2 ± 0.5	doesn't scale	24.5 ± 0.6	> 48 hours	21.1 ± 1.9	67.4 ± 1.1	10.5 ± 0.6	$\mathbf{77.1 \pm 0.6}$
	Coauthor CS	8.6 ± 4.6	76.8 ± 0.6	doesn't scale	19.1 ± 0.5	> 48 hours	64.0 ± 2.0	88.9 ± 0.8	3.8 ± 0.1	$\mathbf{94.0 \pm 0.4}$

Degree Model: In this model we only remove edges. Edges with higher degrees are more likely to be removed to preserve the structure of the graph. Specifically, the probability of removing edge (i, j) is set to $\frac{s_{ij} d_i d_j}{\sum_{ij} s_{ij} d_i d_j}$, where d_i is the degree of node v_i and $s_{i,j}$ is the (i, j) element of the graph adjacency.

We test the performance of `T-GAE` for large-scale network alignment on the degree perturbation model, as described in Section 5.2. We adopt the same setting as in Section 5.2 to train the `T-GAE` according to (6) on small-size networks, i.e., Celegans, Arena, Douban, and Cora, and conduct transfer learning experiments on the larger graphs, i.e., Dblp, and Coauthor CS. The trained T-GAE is used to generate node embedding for the graphs, and Algorithm 1 computes the assignment matrix. The results presented in Table 5 are based on the average and standard deviation of the matching accuracy over 10 randomly generated perturbed samples.

We observe that the benefit of processing the `NetSimile` embeddings with GNNs is still significant in this perturbation model as we observe up to 46% performance increase at the presence of perturbation. When testing on perturbed graphs at 1% level of edge removal, our proposed `T-GAE` consistently outperforms all the competing baselines, while being robust and efficient when performing network alignment under different perturbation models. Especially, on large-scale networks, `T-GAE` is able to achieve very high levels of matching accuracy for both Dblp and Coauthor CS, for $p = 0\%,\ 1\%$.

The benefit of our proposed `T-GAE` framework in improving the expressiveness of GNN still stands out if we compare the accuracy with WAlign and GAE. It also consistently achieves the best result among all baseline methods that are salable.

H Efficiency analysis

H.1 Running time comparison

T-GAE is a scalable and efficient approach for network alignment. We analyze the efficiency of the proposed graph matching framework by comparing its running time with the competing algorithms. T-GAE achieves at most $\times 2000$ less running time, on graph matching tasks, compared to the optimization-based methods, as shown in Table 6. Compared to the existing GNN based approaches, T-GAE consistently achieves the shortest training time and inference time, this is because we replace some message passing layers by the local MLP which serves as an attention function on the output of all GNN layers, such models are empirically proved to be more efficient, at the same time, prompting the expressiveness of GNN layers, to generate node embedding for graph matching. It should be noted that T-GAE is also the only approach that is transferable, which means it does not need to re-train on every pair of new graphs. T-GAE greatly improves the scalability of optimization based methods, as well as the effectiveness of the existing GNN frameworks.

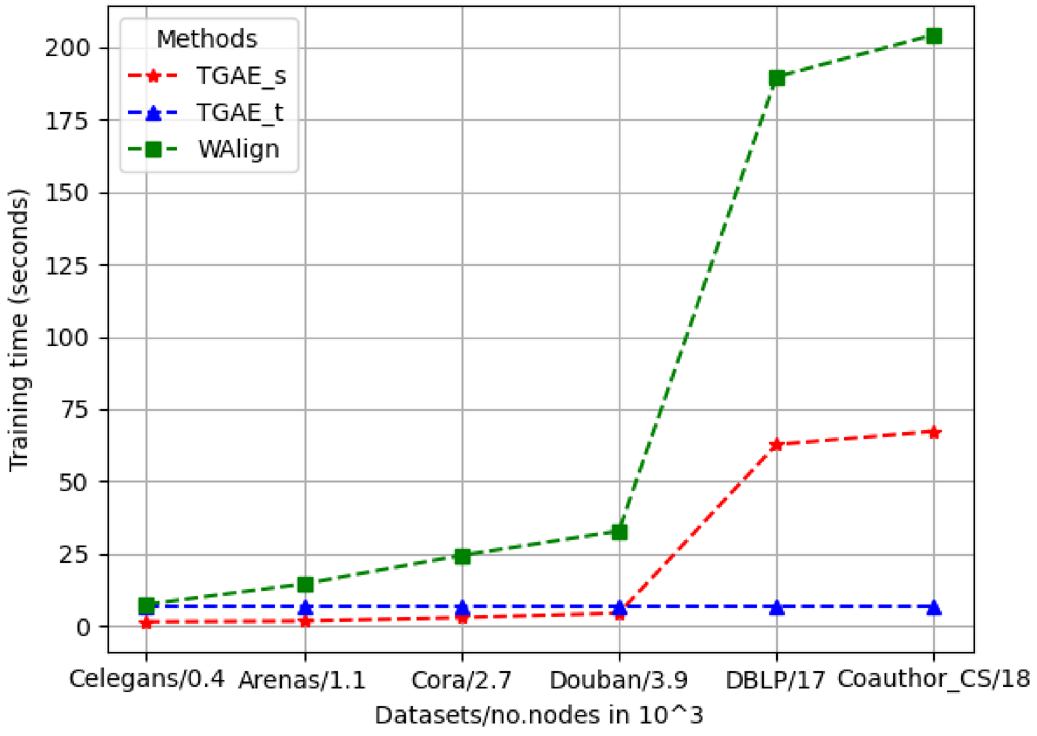

Figure 5: Training time comparison (20 epoches) between T-GAE and WAlign for graph-matching. TGAE_s is the specific setting where we train the encoder GNN according to Equation (7), whereas TGAE_t means training according to Equation (8) on a family of graphs. (Celegans, Arenas, Cora, Douban)

Table 6: Runtime(inference+matching) in seconds for the competing algorithms on graph matching tasks.

Algorithm	Celegans	Arenas	Cora	Douban	Dblp	Coauthor CS
Spectrul	5.712	2.819	54.770	60.298	> 48 hours	> 48 hours
Netsimile	1.212	1.560	1.616	4.400	195.542	225.546
GraphWave	8.308	32.994	131.230	281.629	5470.724	6291.368
FINAL(Matlab)	0.030	0.081	0.498	1.007	86.788	118.065
S-GWL	27.844	37.443	311.201	3394.522	> 48 hours	> 48 hours
ConeAlign	1.333	3.500	13.799	31.955	887.090	1099.145
WAlign	0.078	0.205	0.766	1.800	169.694	189.032
GAE	0.074	0.212	0.757	1.749	164.410	184.062
T-GAE(ours)	**0.068**	**0.201**	**0.742**	**1.734**	**163.836**	**183.289**

The proposed training objective (7) scales well from networks with 400 nodes [Kunegis, 2013] to denser networks with $\times 50$ nodes [Shchur et al., 2018], with minor running time increase, compared to other GNN-based frameworks(WAlign), as shown in Figure 5.

H.2 Matching algorithms comparison

In this subsection, we evaluate the accuracy and matching time of different matching algorithms, to demonstrate how matching algorithms of different time complexity influence the performance of the proposed T-GAE framework. To guarantee fairness of comparison, we use an untrained T-GAE encoder to encode the graphs, and use (1) approximated NN algorithm introduced in Section 4.3 of time complexity $\mathcal{O}(NlogN)$. (2) greedy Hungarian algorithm as applied in Section 5 and 5.3 of time complexity $\mathcal{O}(N^2)$. (3) exact Hungarian algorithm [Kuhn, 1955b] of time complexity $\mathcal{O}(N^3)$, where N is the number of nodes in the graph. We report average accuracy and running time on matching 10 randomly generated samples.

Table 7: Graph matching performance and matching time on Celegans and Arenas using untrained T-GAE encoder. We report results of approximated NN matching algorithm, greedy Hungarian algorithm, and exact Hungarian algorithm. We highlight the performance gain of exact Hungarian over the approximated NN and Greedy Hungarian.

#	Dataset/Perturbation	Celegans						Arenas					
		0		0.01		0.05		0		0.01		0.05	
		Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time
		$\mathcal{O}(NlogN)$											
1	Approximated NN	89.8 ± 1.0	0.004	0.8 ± 0.3	0.004	0.7 ± 0.2	0.004	98.0 ± 2.6	0.009	0.1 ± 0.1	0.009	0.0 ± 0.0	0.009
		$\mathcal{O}(N^2)$											
2	Greedy Hungarian	88.4 ± 0.9	0.068	80.3 ± 0.3	0.068	58.2 ± 3.5	0.067	97.6 ± 0.4	0.173	93.1 ± 0.5	0.176	60.2 ± 5.2	0.174
		$\mathcal{O}(N^3)$											
3	Exact Hungarian	88.5 ± 0.8	0.225	84.0 ± 0.2	24.778	64.7 ± 1.8	73.674	97.6 ± 0.4	0.411	93.8 ± 0.4	562.221	70.0 ± 2.0	1624.713
		$\mathcal{O}(N^3)$											
4	Acc Gain in % (Exact vs Approx/Greedy)	-1.3 / +0.1		+83.2 / +3.7		+64.0 / +6.5		-0.4 / 0		+93.7 / +0.7		+70.0 / +9.8	

The performance of T-GAE to match small graphs can be further improved by adopting the exact Hungarian algorithm. Comparing Exact Hungarian and Greedy Hungarian, we observe that when there is no perturbation involved, greedy Hungarian and exact Hungarian achieves comparable performance. However, the exact Hungarian algorithm outperforms the greedy version in occurrence of perturbations, and the performance gap increases as we introduce more topology noise. Specifically, it offers a 6.5% and 9.8% matching accuracy increase on Celegans and Arenas, respectively, on an untrained T-GAE encoder. However, it can take more than $\times 9000$ longer than the greedy algorithm, on Arenas 5% perturbation, for example. This implies that whenever applicable, the exact Hungarian algorithm should be applied to further improve the performance of T-GAE to match small graphs, especially when the noise level is high, but there is a trade-off between matching accuracy and efficiency.

The efficiency of T-GAE to match permuted graphs can be enhanced by applying the approximated NN algorithm. We observe that the approximated NN algorithm we introduced in Section 4.3 effectively match the aligned nodes of permuted graphs, and on Arenas dataset of 1,133 nodes, it saves 95% matching time compared to the greedy Hungarian algorithm. However, this algorithm fails to match perturbed graphs. This is because the 1-dimensional feature is not expressive enough to to catch the topological noise. Our experiments prove that this efficient NN algorithm should be applied when we match very large scale networks with their permuted versions.

Overall, we divide graph matching using T-GAE in three scenarios: (1) Matching small graphs, exact Hungarian algorithm should be applied. (2) Matching large scale networks without perturbation, the approximation NN algorithm should be deployed to enhance efficiency. (3) The greedy Hungarian algorithm provides a good trade-off between time and efficiency for the general form of graph matching.