Interpretable Neural Subgraph Matching for Graph Retrieval

Abstract

Given a query graph and a database of corpus graphs, a graph retrieval system aims to deliver the most relevant corpus graphs. Graph retrieval based on subgraph matching has a wide variety of applications, *e.g.*, molecular fingerprint detection, circuit design, software analysis, and question answering. In such applications, a corpus graph is relevant to a query graph, if the query graph is (perfectly or approximately) a subgraph of the corpus graph. Existing neural graph retrieval models compare the node or graph embeddings of the query-corpus pairs, to compute the relevance scores between them. However, such models may not provide edge consistency between the query and corpus graphs. Moreover, they predominantly use symmetric relevance scores, which are not appropriate in the context of subgraph matching, since the underlying relevance score in subgraph search should be measured using the partial order induced by subgraph-supergraph relationship. Consequently, they show poor retrieval performance in the context of subgraph matching. In response, we propose ISONET, a novel interpretable neural edge alignment formulation, which is better able to learn the edge-consistent mapping necessary for subgraph matching. ISONET incorporates a new scoring mechanism which enforces an asymmetric relevance score, specifically tailored to subgraph matching. ISONET's design enables it to directly identify the underlying subgraph in a corpus graph, which is relevant to the given query graph. Our experiments on diverse datasets show that ISONET outperforms recent graph retrieval formulations and systems. Additionally, ISONET can provide interpretable alignments between query-corpus graph pairs during inference, despite being trained only using binary relevance labels of whole graphs during training, without any fine-grained ground truth information about node or edge alignments.

1 Introduction

The goal of graph retrieval is to search for the most *relevant* or *similar* graphs, from a corpus graph database, in response to a given query graph. A graph retrieval system must effectively sort corpus graphs based on their relevance scores with respect to a given query graph, and then returns the top-K graphs from this ranked list. Modeling the relevance score between a query and a corpus graph is the central challenge of any graph retrieval system, which predominantly relies on full-graph or subgraph matching, graph kernel design, graph edit distance (GED) and maximum common subgraph (MCS) detection. In this work, we focus on the notion of (approximate) subgraph matching for computing relevance scores, driven by several real-world applications, *e.g.*, molecular fingerprint detection (Cereto-Massagué et al. 2015), circuit design (Ohlrich et al. 1993), object detection (Wong 1992), scene graph retrieval (Johnson et al. 2015), software analysis (Ling et al. 2021), and frequent subgraph mining (Yan, Yu, and Han 2004). Often, an important piece is to find a subgraph in a corpus graph, which is (approximately) isomorphic to the given query graph.

The closely-related problem of full-graph matching has been heavily investigated by the computer vision community (Schellewald and Schnörr 2005; Leordeanu and Hebert 2005; Cho, Lee, and Lee 2010; Bernard, Theobalt, and Moeller 2018; Yu et al. 2018). More recently, trainable neural models have aimed to solve the underlying quadratic assignment using explicit supervision of the ground truth node-level alignments (Zanfir and Sminchisescu 2018; Wang, Yan, and Yang 2019; Tan et al. 2021; Zhao, Tu, and Xu 2021; Fey et al. 2020; Yu et al. 2019). For the applications mentioned earlier, such fine-grained supervision may entail excessive cognitive burden.

1.1 Limitations of prior work

A slew of recent studies proffer neural approaches for graph retrieval. These use a wide variety of relevance measures, *e.g.*, GED, MCS (Li et al. 2019; Bai et al. 2019, 2020; Doan et al. 2021; Peng, Choi, and Xu 2021), complete graph isomorphism (Li et al. 2019), subgraph isomorphism (Lou et al. 2020), *inter alia*. However, they share the following limitations in the context of subgraph matching for graph retrieval.

Node or graph embedding based alignment. Subgraph search should find edge-consistent mappings between query and corpus graphs. However, existing graph retrieval formulations lack explicit edge mapping mechanisms. Instead, they either attempt to align nodes to nodes, depending on contextual embeddings for consistency, or they align aggregated, coarse-grained embeddings of entire graphs to train neural scoring models (Li et al. 2019; Doan et al. 2021). However, without any edge matching protocol, such embeddings are often unable to capture edge correspondences between query-corpus pairs. This leads to inferior performance.

Symmetric scoring function. Most existing graph retrieval models learn a relevance scoring model which is symmetric in query-corpus pairs (Doan et al. 2021; Bai et al. 2019; Li et al. 2019; Bai et al. 2020). This prevalent similarity metric does not naturally lend itself to the subgraph matching task, where there exists a partial ordering induced by the subgraph-supergraph relationship. Lou et al. (2020) partially address this concern. However, they additionally use an explicit annotation of anchor nodes, which may not be explicitly available in real datasets.

Need for fine-grained supervision. Some graph matching formulations (Nowak et al. 2018; Zanfir and Sminchisescu 2018; Wang, Yan, and Yang 2019; Tan et al. 2021; Zhao, Tu, and Xu 2021; Fey et al. 2020; Yu et al. 2019, 2021; Liu et al. 2021; Qu et al. 2021) benefit from direct fine-grained supervision from gold node-to-node matches. Preparing such training data is exceedingly labor-intensive; practical graph retrieval systems may provide only binary relevance labels for query-corpus graph pairs. Ideally, graph retrieval systems should be able to learn relevance functions effectively from such distant supervision.

Lack of interpretability. Ideally, a graph retrieval system should be able to justify its relevance scores with approximate *alignment witnesses*. Many existing systems map each graph to a single aggregated vector, which are compared to estimate similarity scores (Bai et al. 2019; Li et al. 2019; Lou et al. 2020). While such graph-level aggregates can be expressive and accurate, they suffer from a lack of interpretability. Very recently, Doan et al. (2021) attempted to address this concern. However, they used node-based alignment and a symmetric scoring function, which constrain their predictive power in the context of subgraph matching.

1.2 Our Contributions

Responding to the above limitations, we propose ISONET, an interpretable neural graph retrieval model based on subgraph matching. We make the following contributions.

Interpretable edge alignment network. At a high level, ISONET consists of a neural edge alignment network which learns the optimal alignment of the *edges* of a corpus graph with respect to a given query graph. ISONET is built on top of a graph message passing framework, that distills the structural information from local neighborhoods into corresponding node-level and edge-level embeddings. The edge embeddings are subsequently used to learn an optimal alignment plan using a fully differentiable Gumbel-Sinkhorn network (Mena et al. 2018). Such an edge alignment network allows us to approximate the underlying correspondence between query-corpus graph pairs more accurately than the existing node or graph alignment based methods (Doan et al. 2021; Lou et al. 2020; Bai et al. 2019; Li et al. 2019). Moreover, it renders our model interpretable. In addition to accurate retrieval, it pinpoints the subgraph within a relevant corpus graph, which is approximately isomorphic to the query graph.

Asymmetric loss. The goal of the edge alignment network is to maximize the number of query edges that are *covered* by a strongly similar corpus edge. This is an asymmetric target, in sharp contrast with symmetric goals like minimizing the Frobenius difference between the query edge embeddings and permuted corpus edge embeddings, or maximizing the number or weight of matched edges. ISONET applies a suitable hinge loss to approximate the number of uncovered query edges. Thus, unmatched components of the corpus graph do not impact the relevance score. This is clearly better suited for subgraph search, and is evident from experiments.

Distant supervision. ISONET can be effectively trained using a pairwise ranking loss under the supervision of binary whole-graph relevance indicators between query-corpus pairs, without any direct supervision of node or edge alignments.

Comprehensive evaluation. ISONET outperforms several state-of-the-art graph retrieval models by a significant margin on six diverse datasets. Anecdotal evidence suggests that approximate alignments can be "read out" at test time from the internal state of ISONET, and these are structurally consistent, despite only distant supervision at training time.

2 Preliminaries

In this section, we set up notation, provide the key components of graph retrieval system and finally, briefly describe our broad goal in this paper.

2.1 Notation

We are provided a set of (undirected) graphs $D = Q \cup C$, where $Q = \{G_q = (V_q, E_q) \,|\, q \in [|Q|]\}$ is the set of query graphs; $C = \{G_c = (V_c, E_c) \,|\, c \in [|C|]\}$ is a set of corpus graphs. Moreover, for every query graph G_q, we are given the set of relevant graphs $C_{q\oplus} \subset C$ and a set of irrelevant graphs $C_{q\ominus} \subset C$, formalized as follows.

$$C_{q\oplus} = C \backslash C_{q\ominus} = \{G_c \in C : G_q \,\text{is relevant to}\, G_c\}. \quad (1)$$

In this context, we define the relevance label $y(G_q, G_c) = 1$ if $G_c \in C_{q\oplus}$ and 0, otherwise. We use $G = (V, E)$ to denote any graph from $Q \cup C$; u_q, v_q and u_c, v_c to denote the nodes of a query graph G_q and a corpus graph G_c respectively; and, u, v to denote the nodes of any graph $G \in Q \cup C$. We use $\boldsymbol{A}_q$, $\boldsymbol{A}_c$ and $\boldsymbol{A}$ to denote the adjacency matrix of graphs G_q, G_c and G respectively. A permutation of nodes in a graph is denoted $\boldsymbol{S}$, whereas a permutation of edges is denoted by $\boldsymbol{P}$. $\vec{1}$ denotes a tensor of all-1s. Finally, $\text{vec}(\boldsymbol{A})$ denotes the vector obtained by column-wise concatenation of $\boldsymbol{A}$ and $\otimes$ denotes the matrix Kronecker product operation.

2.2 Overview of a graph retrieval system

Given a query graph $G_q \in Q$ and a set of corpus graphs C, a graph retrieval system returns a ranked list of graphs, which are likely to be relevant to G_q. Therefore, we can view a graph retrieval task as an instance of ranking problem. Similar to other information retrieval algorithms, a graph retrieval algorithm works in two steps. First, it computes a relevance score $s(G_c|G_q)$ (or equivalently, a distance $d(G_c|G_q)$ inversely related to relevance). Next, it provides a ranking of C in decreasing (increasing) order of relevance (distance) scores. A graph retrieval algorithm is considered high quality, if the most relevant graphs appear at top ranks. Therefore, optimal design of $s(G_c|G_q)$ or $d(G_c|G_q)$ is the central challenge of any graph retrieval algorithm. In this work, we use distance scoring function $d(G_c \,|\, G_q)$ as the relevance measure, where the corpus graph G_c with the smallest distance w.r.t. G_q has the highest relevance score and is ranked at the top.

2.3 Our goal

Usually, s or d must measure some structural similarity. E.g., in question answering, a dependency graph inferred from the question may be a subgraph of the dependency graph inferred from an answer sentence. In image search, query objects and their relations ("man feeding a dog") may be expected to form a subgraph of a response image with other objects in it. Therefore, we use the notion of subgraph isomorphism to define relevance between two graphs. More formally, we are given a set of query $Q = \{G_q\}$ and corpus graphs $C = \{G_c\}$ along with their relevance labels $y(G_q, G_c)$. Our goal is to design a neural distance function $d(G_c|G_q)$ which can accurately predict the corresponding value of $y(G_q, G_c)$, or rank in strong agreement with $y(G_q, G_c)$.

3 The design of ISONET

A corpus graph G_c can be regarded as having "perfect score" with respect to a query graph G_q if G_c contains a subgraph that is isomorphic to G_q, which is the well-known NP-complete subgraph isomorphism[1] problem. The NP-hardness of its close cousin, graph isomorphism[2], has been open for at least 40 years. Our goals are related only to continuous relaxations of these combinatorial problems, to make it amenable to graph retrieval task. Because we want plausible matches to nodes and edges with modified or distorted features and neighborhoods, soft-matching is, in fact, a desired feature. We will begin by presenting a view of graph search based on node alignments, then extend it to include an edge-centric view to complete the design of our full system, ISONET.

3.1 Node alignment approach and its limitations

We develop a distance score using node alignment and illustrate its limitations, which we fix in the next section.

Graph isomorphism test using node alignment. Let V_q, V_c be the set of nodes of G_q, G_c. Generally $|V_q| < |V_c|$. We pad G_q with $|V_c| - |V_q|$ disconnected dummy nodes. In their augmented forms, let $\boldsymbol{A}_q$ and $\boldsymbol{A}_c$ be the node adjacency matrices (of the same size N). Consider a node permutation matrix $\boldsymbol{S}$ of the same size as $\boldsymbol{A}_q$ and $\boldsymbol{A}_c$. Then $\boldsymbol{S}\boldsymbol{A}_c\boldsymbol{S}^\top$ is the row- and column-permuted transformation of $\boldsymbol{A}_c$. In case of the isomorphism test, a suitable objective would be (Grohe, Rattan, and Woeginger 2018):

$$\operatorname{argmin}_{\boldsymbol{S}} \textstyle\sum_{i,j} \left\| \boldsymbol{A}_q - \boldsymbol{S}\boldsymbol{A}_c\boldsymbol{S}^\top \right\|_F^2 . \qquad (2)$$

Seeking node alignment for edge coverage. The case of subgraph isomorphism is somewhat different. If $G_q \subseteq G_c$, then there exists a permutation $\boldsymbol{S}$ such that each non-zero element position in $\boldsymbol{A}_q$ is also a non-zero element in $\boldsymbol{S}\boldsymbol{A}_c\boldsymbol{S}^\top$. In other words, the subgraph isomorphism test can be accomplished by optimizing the following objective:

$$\operatorname{argmin}_{\boldsymbol{S}} \textstyle\sum_{i,j} \left[(\boldsymbol{A}_q - \boldsymbol{S}\boldsymbol{A}_c\boldsymbol{S}^\top)_+ \right]_{i,j} , \qquad (3)$$

where $(\bullet)_+ = \max\{0, \bullet\}$ is the ReLU/hinge function. Any $\boldsymbol{S}$ that reduces the objective to zero certifies a subgraph isomorphism, but the objective can be used as a relevance score for how close G_c comes to containing G_q.

[1]https://en.wikipedia.org/wiki/Subgraph_isomorphism_problem
[2]https://en.wikipedia.org/wiki/Graph_isomorphism_problem

Approximation of (3) **with node embeddings.** In practical graph search applications, nodes and edges may be characterized by continuous and noisy features. A chlorine atom (or a word like 'cat') in a query molecule (or sentence parse) graph may approximately match a functionally similar halide like bromine (or the word 'panther') in a corpus graph. To capture such node-to-node similarities, node embeddings are commonly used. Let $\boldsymbol{h}_u \in \mathbb{R}^D$ denote the embedding of node u, and these be collected into the node embedding matrix $\boldsymbol{H}_q, \boldsymbol{H}_c \in \mathbb{R}^{N\times D}$. If there exists an $\boldsymbol{S}$ such that $\boldsymbol{H}_q \approx \boldsymbol{S}\boldsymbol{H}_c$, that reflects symmetric similarity. But if, as in (3), we wish to detect coverage of query by corpus, we want $\boldsymbol{H}_q \le \boldsymbol{S}\boldsymbol{H}_c$ (elementwise). Such "order embedding" constraints have been used in entailment between WordNet synsets (Vendrov et al. 2015) and whole sentences (Lai and Hockenmaier 2017). This suggests a natural (asymmetric) distance function as the relevance measure:

$$d(G_c|G_q) = \min_{\boldsymbol{S}} \textstyle\sum_{i,j} \left[(\boldsymbol{H}_q - \boldsymbol{S}\boldsymbol{H}_c)_+ \right]_{i,j}, \qquad (4)$$

where G_c with the smallest distance has the highest relevance score to G_q and, is ranked at the top.

Limitations. The node alignment approach suffers from the following limitations.

- In the context of graph retrieval, $d(G_q \,|\, G_c)$ are fed into a ranking objective during training. Hence, we need to solve the minimization problem (4) as a part of end-to-end training, which in turn demands a differentiable routine to compute the optimal node permutation $\boldsymbol{S}$. However, differentiable neural gadgets mostly aim to solve linear assignment problems (Cuturi 2013; Mena et al. 2018), whereas the optimization (4) is a notoriously hard quadratic assignment problem (also rendered somewhat different in nature here, thanks to the ReLU term).
- For (4) to provide a reasonable approximation to (3) and provide edge agreement between nodes determined to be corresponding, the embedding of a node u has to encapsulate neighborhood information. In other words, these node embeddings must be *contextual* (Kipf and Welling 2016; Hamilton, Ying, and Leskovec 2017; Xu et al. 2018; Veličković et al. 2017). Despite these measures, our experiments suggested that edge subsumption had to be tackled directly, rather than depend on contextual node embeddings. We develop this idea next.

3.2 From node to edge alignment+coverage

Here, we give an edge alignment based representation of the subgraph matching objective in Eq. (4) which ameliorates the above limitations.

Subgraph matching with edge alignment. Part of the notorious hardness of subgraph isomorphism comes from the need for node permutations to honor edge correspondence or subsumption. To tackle this problem, we directly enforce edge correspondence in our retrieval model. We motivate this by rewriting the objective of (3) as

$$\begin{aligned} &\sum_{i,j\in[N]\times[N]} \left[(\boldsymbol{A}_q - \boldsymbol{S}\boldsymbol{A}_c\boldsymbol{S}^\top)_+ \right]_{i,j} \\ &\quad = \sum_{k\in[N^2]} \Big[\big(\operatorname{vec}(\boldsymbol{A}_q) - (\boldsymbol{S}\otimes\boldsymbol{S}) \operatorname{vec}(\boldsymbol{A}_c) \big)_+ \Big]_k \qquad (5) \end{aligned}$$

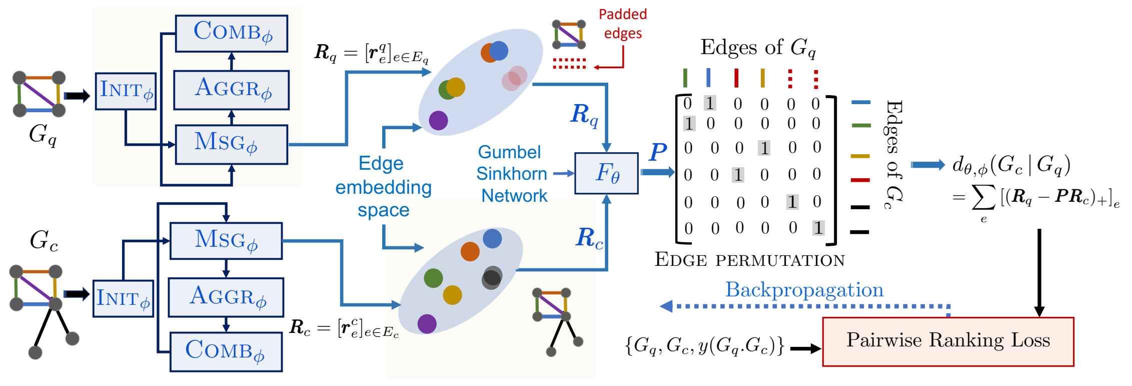

Figure 1: Neural architecture of IsoNet. From left to right, given a query-corpus graph pair (G_q, G_c), the graph neural network $\{\textsc{Init}_\phi, \textsc{Msg}_\phi, \textsc{Aggr}_\phi, \textsc{Comb}_\phi\}$ learns the context sensitive node $\boldsymbol{H} = [\boldsymbol{h}_u]$ and edge $\boldsymbol{R} = [\boldsymbol{r}_e]$ representations. These embeddings encapsulate structural information from the local neighborhood. We pad $\boldsymbol{R}_q$ with $|E_c| - |E_q|$ zero vectors which correspond to dummy edges. Next we feed $(\boldsymbol{R}_q, \boldsymbol{R}_c)$ into the permutation generator F_θ which is designed using a Gumbel-Sinkhorn neural network. This network outputs an approximate edge alignment denoted by matrix $\boldsymbol{P}$. Finally, we employ a asymmetric relevance score with the distance $d_{\theta,\phi}(G_c \mid G_q)$ using the pair of aligned edge embeddings $(\boldsymbol{R}_q, \boldsymbol{P}\boldsymbol{R}_c)$. Given a set of query and corpus graph pairs $\{(G_q, G_c) \mid q \in Q, c \in C\}$, the relevance measures $\{d_{\theta,\phi}(G_c \mid G_q)\}$ are used to train the underlying parameter θ, against ground truth binary relevance labels, by minimizing a pairwise ranking loss.

Here, N is the total number of nodes in the augmented graph. Note that, the Kronecker product $\boldsymbol{S} \otimes \boldsymbol{S}$ can be established to be a permutation matrix itself, because $(\boldsymbol{S} \otimes \boldsymbol{S})\vec{1}_{N\times N} = (\boldsymbol{S} \otimes \boldsymbol{S})(\vec{1}_N \otimes \vec{1}_N) = (\boldsymbol{S}\vec{1}_N)(\boldsymbol{S}\vec{1}_N) = \vec{1}_N \otimes \vec{1}_N = \vec{1}_{N^2}$. On the other hand, $\text{vec}(\boldsymbol{A}_\bullet)$ in Eq. (5) is nothing but a vector of edge presence indicators over all N^2 edge slots.

Approximating (5) **with edge embeddings.** Translating back to the world of embedding representations, we use edge rather than node embeddings, which we denote as $\boldsymbol{r}_e$ for edge e and $\boldsymbol{R}_q, \boldsymbol{R}_c \in \mathbb{R}^{|E|\times D}$ collectively.[3] We can thus write our distance measure using edge embeddings, similar to (4), as

$$d(G_c \mid G_q) = \min_{\boldsymbol{P}} \textstyle\sum_e [(\boldsymbol{R}_q - \boldsymbol{P}\boldsymbol{R}_c)_+]_e, \qquad (6)$$

where $\boldsymbol{P}$ is now an *edge* permutation. In terms of optimization, the potential advantage of (6) over (3) is that we avoid a quadratic term in $\boldsymbol{P}$, in contrast to the quadratic term involving $\boldsymbol{S}$.

3.3 IsoNet architecture

Here, we develop our retrieval model by building upon the edge alignment based representation, described above. We approximate $d(G_c \mid G_q)$ in Eq. (6) with $d_{\theta,\phi}(G_c \mid G_q)$ using two neural networks— where the first network with parameter θ aims to provide a differentiable solution $\boldsymbol{P}^*$ for the underlying optimization problem, and the second network with parameter ϕ models the edge representation matrices $\boldsymbol{R}_q$ and $\boldsymbol{R}_c$.

Design of the network to compute $\boldsymbol{P}^*$. The matrix $\boldsymbol{P}$ in Eq. (6) is a 'hard' permutation matrix. The next step is to relax $\boldsymbol{P}$ to a doubly stochastic or 'soft' permutation matrix, which will be the output of a suitable network $F_\theta(\boldsymbol{R}_q, \boldsymbol{R}_c)$, with trainable parameters θ. In other words, the combinatorial search for discrete $\boldsymbol{P}$ is replaced with the relaxation

$$d_{\theta,\phi}(G_c \mid G_q) = \textstyle\sum_{i,j} \Big[\big(\boldsymbol{R}_q - F_\theta(\boldsymbol{R}_q, \boldsymbol{R}_c)\boldsymbol{R}_c\big)_+\Big]_{i,j} \qquad (7)$$

As shown in the seminal work by Cuturi (2013), such an approach is compatible with a linear assignment problem. The optimization (6) is still not close to a linear assignment problem, due to the presence of the ReLU() term. However, in the following, we show that it can be regarded as a linear assignment problem in the dual space without the interference from a hinge operator. Specifically, we write the dual of the minimization problem in Eq (6) as (See Appendix B for details):

$$\max_{\boldsymbol{C}\in[0,1]} \min_{\boldsymbol{P}} \text{Trace}\left[\boldsymbol{C}^\top (\boldsymbol{R}_q - \boldsymbol{P}\boldsymbol{R}_c)\right] \qquad (8)$$

where $\boldsymbol{C}_{i,j}$ are the Lagrangian multipliers. Now, suppose we can compute the optimal $\boldsymbol{C}^* = \boldsymbol{C}^*(\boldsymbol{R}_q, \boldsymbol{R}_c)$, then the inner minimization problem becomes equivalent to:

$$\min_{\boldsymbol{P}} \text{Trace}[-\boldsymbol{P}^\top \boldsymbol{C}^*(\boldsymbol{R}_q, \boldsymbol{R}_c)\boldsymbol{R}_c^\top]. \qquad (9)$$

Thus, the above optimization is a linear assignment problem with cost matrix $\boldsymbol{C}^*(\boldsymbol{R}_q, \boldsymbol{R}_c)\boldsymbol{R}_c^\top$. To approximate this term, we first pass each of $\boldsymbol{R}_q, \boldsymbol{R}_c$ separately through a linear-ReLU-linear network 'LRL' with tied parameters and then perform an inner product to find raw pairwise similarities. Next, these similarities inform a (soft) permutation finder network, for which we use the differentiable Gumbel-Sinkhorn operator 'GS' (Mena et al. 2018):

$$F_\theta(\boldsymbol{R}_q, \boldsymbol{R}_c) = \text{GS}\left(\text{LRL}_\theta(\boldsymbol{R}_q) \cdot \text{LRL}_\theta(\boldsymbol{R}_c)^\top\right) \qquad (10)$$

The role of GS is to find an approximate solution of the linear assignment problem whose cost matrix is fed as input to it. Thus, in our setup, F_θ approximates $\boldsymbol{P}^*$, the optimal permutation matrix which solves (9). In more detail, the GS network performs alternating row and column scaling on the input matrix $\boldsymbol{M}$ after an initial transformation with a

[3]Node and edge embeddings need not both be D-dimensional, but we reuse D to economize on notation.

temperature $\tau > 0$:

$$\mathrm{GS}(\boldsymbol{M}) := \lim_{t\to\infty} \mathrm{GS}^t(\boldsymbol{M}), \text{ where} \tag{11}$$

$$\mathrm{GS}^0(\boldsymbol{M}) = \exp(\boldsymbol{M}/\tau) \quad \text{and} \tag{12}$$

$$\mathrm{GS}^t(\boldsymbol{M}) = \mathrm{ColScale}\Big(\mathrm{RowScale}\big(\mathrm{GS}^{t-1}(\boldsymbol{M})\big)\Big) \tag{13}$$

It is known (Mena et al. 2018) that $\mathrm{GS}(\boldsymbol{M})$ is the solution to the optimization

$$\max_{\boldsymbol{P}\in\mathcal{B}} \mathrm{Trace}(\boldsymbol{P}^\top \boldsymbol{M}) - \tau \textstyle\sum_{i,j} \boldsymbol{P}_{i,j} \log \boldsymbol{P}_{i,j}, \tag{14}$$

where $\mathcal{B}$ is the set of doubly stochastic matrices. In other words, leaving aside the entropy term (which is annealed toward 0 using τ), $\mathrm{GS}(\boldsymbol{M})$ tends to become a hard permutation matrix.

Designing neural network for edge embeddings $\boldsymbol{R}_\bullet$. It remains to specify the computation of edge embedding matrices $\boldsymbol{R}_q, \boldsymbol{R}_c$. We use a message passing framework for learning context sensitive node embeddings which are then used to compute the edge embeddings $\boldsymbol{R}_\bullet$ (Gilmer et al. 2017). First, for each node $u \in V$, we map the input node feature x_u to an initial node embedding as:

$$\boldsymbol{h}_u(0) = \textsc{Init}_\phi(\boldsymbol{x}_u) \tag{15}$$

Next, given an integer K, a recurrent propagation layer is used to aggregate structural information from the K-hop subgraph centered around each node. Each individual propagation step, for $k \le K$, involves computing a message signal for every edge $(u, v) \in E$, and subsequently updating each node embedding using an aggregation of its neighboring messages.

$$\boldsymbol{r}_{(v,u)}(k-1) = \textsc{Msg}_\phi([\boldsymbol{h}_v(k-1); \boldsymbol{h}_u(k-1)]) \tag{16}$$

$$\overline{\boldsymbol{r}}_u(k-1) = \textsc{Aggr}_\phi\left(\{\boldsymbol{r}_{(v,u)}(k-1) \mid v\in \mathrm{nbr}(u)\}\right) \tag{17}$$

$$\boldsymbol{h}_u(k) = \textsc{Comb}_\phi\left(\boldsymbol{h}_u(k-1), \overline{\boldsymbol{r}}_u(k-1)\right) \tag{18}$$

The trainable parameters of all the above networks are collectively denoted as ϕ. Finally, the contextual edge embedding matrix becomes $\boldsymbol{R} = [\boldsymbol{r}_{(u,v)}(K)]_{(u,v)\in E}$. Following Li et al. (2019), we use two multilayer perceptrons (MLP) for the initial embedding computation network INIT and the message passing network MSG. While AGGR can be any symmetric operator, we use a simple sum in this work. Finally, we use a gated recurrent unit (GRU) for COMB, following the proposal of Li et al. (2015). Appendix D contains more details about the neural architecture of ISONET.

3.4 Scoring, ranking loss, and training

A salient feature of ISONET is that we do not expect detailed supervision in the form of gold (sub)isomorphisms between G_q and G_c for relevant corpus graphs. Instead, our instances comprise a query graph with limited sets $C_{q\oplus}, C_{q\ominus}$ of relevant and irrelevant corpus graphs, respectively. Following a long line of work in learning to rank (Joachims 2002; Liu 2009), we use a pairwise ranking loss. If $G_{c\oplus}$ and $G_{c\ominus}$ are relevant and irrelevant graphs for query graph G_q, then we want

$$d_{\theta,\phi}(G_{c\oplus}|G_q) \ll d_{\theta,\phi}(G_{c\ominus}|G_q), \tag{19}$$

where subscripts θ, ϕ reflect the design of the scoring function through previous sections. We implement the overall training of θ through a hinge loss with a margin hyperparameter $\gamma > 0$:

$$\min_{\theta,\phi} \sum_{G_q} \sum_{\substack{G_{c\oplus}\\ G_{c\ominus}}} \left[\gamma + d_{\theta,\phi}(G_{c\oplus}|G_q) - d_{\theta,\phi}(G_{c\ominus}|G_q)\right]_+ \tag{20}$$

For clarity, this hinge loss is unrelated to the hinge used to assess coverage of G_q edges by edges from G_c.

4 Experiments

In this section, we evaluate our proposals across six real datasets. Specifically, we aim to answer the following research questions. **RQ1:** How does ISONET perform as compared to the existing state-of-the-art graph retrieval models? **RQ2:** How much performance gain is brought about by the edge alignment network in our model, as compared to its node alignment counterpart? **RQ3:** Is the asymmetric scoring model useful? How much of an accuracy boost does it give when compared to a symmetric scoring model? **RQ4:** Are the scores provided by ISONET supported by interpretable alignments? Can ISONET identify the hidden subgraph in a relevant corpus graph, which is isomorphic to the given query graph? Appendix E contains additional experiments.

4.1 Experimental setup

Datasets. We experiment with six real world datasets: PTC-FR, PTC-FM, PTC-MM, PTC-MR, MUTAG and AIDS (Morris et al. 2020). From each dataset, we extract the query-corpus graph pairs as in Lou et al. (2020). We summarize datasets and their preparation in Appendix C.

Baselines. We compare ISONET against eight competitive baselines. They include two graph kernel based methods, *viz.*, (i) random walk kernel (RWKernel), and (ii) shortest path kernel (SPKernel); two graph retrieval models which compare node embeddings in order to compute the relevance scores, *viz.*, (iii) GraphSim (Bai et al. 2020), (iv) GOTSim (Doan et al. 2021) four retrieval models which compare whole graph embeddings to compute the relevance scores, *viz.*, (v) SimGNN (Bai et al. 2019), (vi) GMN-embed (Li et al. 2019), (vii) GMN-match (Li et al. 2019), and (viii) Neuromatch (Lou et al. 2020). GMN-embed computes the query (corpus) graph embedding independently of the corpus (query) graph. In contrast, GMN-match computes the graph embedding using a cross attention layer across the nodes of the query corpus pairs. Thus the underlying query embedding depends on the corresponding corpus graph and vice-versa. Appendix D contains the implementation details of the baselines.

Evaluation protocol. Given a set of query graphs Q and a set of corpus graphs C, we split Q into 60% training, 15% validation and 25% test folds. We train ISONET and the baselines using the training set and select the corresponding hyperparameters using the validation set. Once the model is trained, then for each query G_q in the test set, we rank the corpus graphs G_c by increasing $d(G_c|G_q)$. Next, we compute the average precision (AP) and reciprocal rank (RR) for each of these ranked lists. Finally, we compute the average of both these metrics over the query graphs in the test set, to report

	Mean Average Precision (MAP)						Mean Reciprocal Rank (MRR)					
	PTC-FR	PTC-FM	PTC-MM	PTC-MR	MUTAG	AIDS	PTC-FR	PTC-FM	PTC-MM	PTC-MR	MUTAG	AIDS
SPKernel	0.339	0.372	0.343	0.320	0.430	0.356	0.556	0.574	0.476	0.526	0.751	0.681
RWKernel	0.186	0.214	0.220	0.191	0.183	0.160	0.365	0.414	0.520	0.435	0.147	0.481
GraphSim	0.399	0.359	0.411	0.442	0.324	0.320	0.898	0.549	0.798	0.832	0.686	0.568
GOTSim	0.419	0.553	0.448	0.526	0.631	0.502	0.760	0.751	0.580	0.876	0.828	0.848
SimGNN	0.345	0.539	0.392	0.419	0.548	0.321	0.683	0.773	0.803	0.810	0.797	0.721
GMN-embed	0.739	0.754	0.760	0.766	0.893	0.793	**0.960**	0.841	0.901	0.933	**0.947**	**1.000**
GMN-match	0.774	0.783	0.837	0.805	0.909	0.835	0.953	0.920	0.946	0.960	0.933	0.953
NeuroMatch	0.675	0.686	0.738	0.605	0.841	0.762	0.953	0.933	0.935	0.817	0.913	0.893
IsoNet	**0.857**	**0.901**	**0.906**	**0.894**	**0.920**	**0.894**	**0.960**	**1.000**	**0.980**	**0.970**	0.940	**1.000**

Table 2: Retrieval performance measured in terms of mean average precision (MAP) (left half) and mean reciprocal rank (MRR) (right half) of IsoNet and all the state-of-the-art baselines, *viz.*, Shortest path kernel (SPKernel), Random Walk kernel (RWKernel) (Vishwanathan et al. 2010), GraphSim (Bai et al. 2020), GOTSim (Doan et al. 2021), SimGNN (Bai et al. 2019), GMN-embed, GMN-match (Li et al. 2019) and Neuromatch (Lou et al. 2020), on 25% test set. Among these methods, GraphSim and GOTSim use node alignment methods and, SimGNN, GMN-embed and GMN-match compare graph embeddings to compute relevance scores. Numbers in **bold** (underline) indicate the best (second best) performers. We observe that in most of the cases, IsoNet outperforms recent competitive approaches.

mean average precision (MAP) and mean reciprocal rank (MRR).

4.2 Results

Comparison with baselines (RQ1). Table 2 compares IsoNet against recent state-of-the-art systems across all data sets. We make the following observations.
1) In terms of MAP, IsoNet outperforms all baselines by a substantial margin in all the datasets. In terms of MRR, IsoNet is the best performer in five datasets, except MUTAG, where GMN-embed beats it by $<1\%$ margin.
2) GMN-match is the consistent runner-up in terms of MAP. It offers a significant accuracy boost over other baselines, owing to its cross attention layer, resulting in query-specific corpus embeddings and vice-versa.
3) GMN-embed is consistently good across all datasets, tying with IsoNet's MRR for PTC-FR and AIDS datasets, owing to its highly expressive network compared to other GNN baselines.
4) Similar to GMN-embed, Neuromatch also shows good performance across most datasets. While it is specifically designed for subrgaph matching, it still performs worse than GMN-embed and GMN-match, likely because its underlying graph embedding model is weaker.
5) The performance of SimGNN, GOTSim and GraphSim are poor as compared to the GMNs and Neuromatch. These models use node or graph embeddings which are fed into a symmetric scoring model. Moreover, they use a vanilla GNN, which constrains the ability of the embedding to capture structural information from the graph.
6) Finally, the graph kernel methods show extremely poor performance. This is due to the fact that, they use simple heuristic based methods for computing relevance scores, which are not customized for subgraph based retrieval task.

Edge alignment vs. node alignment (RQ2). To tease out the benefits of direct edge coverage scoring, we design two alternatives of our model, where we replace our edge-alignement network with the corresponding node-alignment counterparts:

	PTC-FR	PTC-FM	PTC-MM	PTC-MR	MUTAG	AIDS
Node-align (Node loss)	0.832	0.859	0.862	0.838	0.886	0.849
Node-align (Edge loss)	0.803	0.847	0.828	0.814	0.815	0.869
IsoNet	**0.857**	**0.901**	**0.906**	**0.894**	**0.920**	**0.894**

Table 3: Performance comparison of IsoNet against its two alternatives: **Node-align (Node loss)** and **Node-align (Edge loss)**, that are obtained by replacing its edge alignment network with two node alignment networks, across all datasets. In the first alternative, we compute the score $d_{\theta,\phi}(G_c \mid G_q) = \sum_{i,j}[(\boldsymbol{H}_q - \boldsymbol{S}\boldsymbol{H}_c)_+]_{i,j}$. In the second alternative, we compute $d_{\theta,\phi}(G_c \mid G_q) = \sum_e[(\boldsymbol{L}_q - \boldsymbol{S}\boldsymbol{L}_c\boldsymbol{S}^\top)_+]_e$, where $\boldsymbol{L}(e) = \boldsymbol{A}(e)\cdot[\text{FF}(\boldsymbol{r}_e)]$.

Node-align (Node loss). In this first alternative, we compute the relevance score as, $d_{\theta,\phi}(G_c \mid G_q) = \sum_{i,j}[(\boldsymbol{H}_q - \boldsymbol{S}\boldsymbol{H}_c)_+]_{i,j}$, where $\boldsymbol{S}$ is the node permutation matrix, the counterpart of $\boldsymbol{P}$ in the node alignment setting. Here, the distance function measures the amount of discrepancy in node correspondence.

Node-align (Edge loss). Here, we compute the relevance score as: $d_{\theta,\phi}(G_c \mid G_q) = \sum_e[(\boldsymbol{L}_q - \boldsymbol{S}\boldsymbol{L}_c\boldsymbol{S}^\top)_+]_e$, where $\boldsymbol{L}(e) = \boldsymbol{A}(e)\cdot[\text{FF}(\boldsymbol{r}_e)]$. Here, the feedforward layer FF maps the edge embedding to a single value. In this setup, we aim to enforce edge correspondence using node alignment as the control variable.

Table 3 summarizes a comparative analysis between IsoNet and the above node alignment models. We make the following observations. (1) IsoNet outperforms both the alternatives, which shows the efficacy of edge alignment network over node alignment network. (2) Despite the underlying edge consistency mechanism, Node-align (Edge loss) is outperformed by Node-align (Node loss). In Node-align (Edge loss), the gradient feedback received by the training algorithm from the last layer is too distant from the underlying permutation network F_θ. Here, F_θ uses node embeddings $\boldsymbol{H}_q$ and $\boldsymbol{H}_c$ to compute $\boldsymbol{S}$, whereas the relevance measure $d_{\theta,\phi}(G_c \mid G_q)$

	ISONET		GMN-embed		GMN-match	
	Symm	Asym	Symm	Asym	Symm	Asym
PTC-FR	**0.858**	0.857	0.739	0.776	0.774	0.786
PTC-FM	0.839	**0.901**	0.754	0.819	0.783	0.806
PTC-MM	0.867	**0.906**	0.760	0.801	0.837	0.845
PTC-MR	0.804	**0.894**	0.766	0.796	0.805	0.807
MUTAG	0.891	**0.920**	0.893	0.897	0.909	**0.920**
AIDS	0.884	**0.894**	0.793	0.836	0.835	0.858

Table 4: Performance comparison between asymmetric (7) and symmetric scoring module, measured using MAP - for ISONET, GMN-match and GMN-embed, across all datasets. Numbers in bold indicate the best performer.

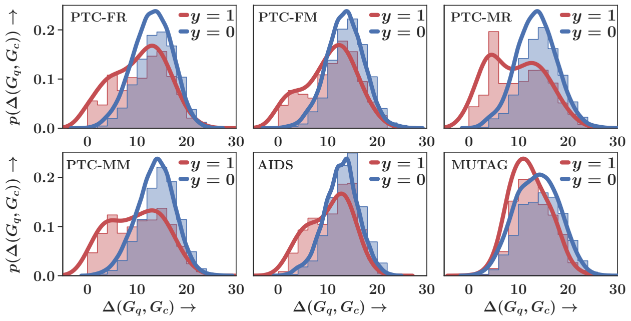

Figure 5: Distributions of the number of node consistency violations $\Delta(G_q, G_c)$ (Eq. (21)) for proposed edge alignments. Here we present the distribution of violations across relevant (red) and irrelevant (blue) query corpus graph pairs.

does not explicitly use $\boldsymbol{H}_q$ and $\boldsymbol{H}_c$ in this case. Such a setup affects end-to-end training of the node embedding model and the permutation network. On the other hand, in Node-align (Node loss), the scoring function explicitly uses the node embeddings, which facilitates end-to-end training.

Asymmetric vs. symmetric score (RQ3). In order to validate the effectiveness of our asymmetric scoring formulation, we conduct two complementary experiments.

- One one hand, we replace the asymmetric scoring function in Eq. (7) with a symmetric alternative $\|\boldsymbol{R}_q - F_\theta(\boldsymbol{R}_q, \boldsymbol{R}_c)\boldsymbol{R}_c\|^2$, similar to the function used for GMN (Li et al. 2019).
- On the other hand, we replace the symmetric metric involving graph level embeddings in GMN-embed and GMN-match with our asymmetric measure.

Table 4 shows the results. We make two observations.

1) Across all three methods, the asymmetric scoring model significantly outperforms the symmetric model, thus clearly exhibiting superior utility for the subgraph matching task.

2) The asymmetric variations of our closest competitors, GMN-embed and GMN-match, are still outperformed by symmetric ISONET in 4 out of the 6 datasets.

This demonstrates that ISONET's edge alignment mechanism holds clear dominance over the graph level representation techniques of GMN, so much so that even when ISONET is handicapped by symmetric scoring and GMN baselines afforded the benefit of asymmetric scoring, ISONET still prevails.

Interpretability analysis (RQ4). We investigate the quality of edge alignments proposed by ISONET. To do so, we measure the extent to which ISONET is able to achieve consistency across all the query-corpus graph pairs. Specifically, we use the following node inconsistency measure:

$$\Delta(G_q, G_c) = \textstyle\sum_{e \neq e'} [(\boldsymbol{I}_q \boldsymbol{I}_q^\top - \boldsymbol{P}(\boldsymbol{I}_c \boldsymbol{I}_c^\top)\boldsymbol{P}^\top)_+]_{e,e'} \quad (21)$$

Here $\boldsymbol{I}_q, \boldsymbol{I}_c$ represent edge-to-node incidence matrices with G_q being padded with $|E_c| - |E_q|$ dummy edges. Therefore, the off diagonal entries in $\boldsymbol{I}\boldsymbol{I}^\top$ indicate whether the corresponding edges are incident on a common node. Thus $\Delta(G_q, G_c)$ in Eq. (21) counts the number of violations where the adjacent edges on the query side are not mapped to adjacent edges on the corpus side. Figure 5 summarizes the distributions of $\Delta(G_q, G_c)$ across all datasets, which shows that the distribution for relevant pairs is noticeably skewed towards zero violations as compared to the distribution for irrelevant pairs. Thus, ISONET achieves significantly greater success at minimizing violations, by finding lower-loss edge permutations more often for relevant graph pairs as compared to irrelevant pairs. Appendix E visualizes how our method is able to find the correct edge alignments.

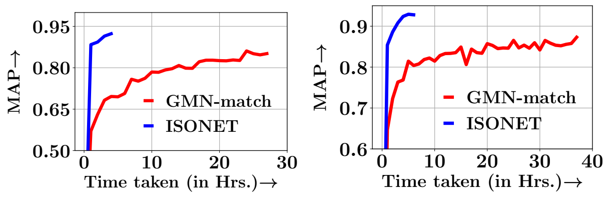

Figure 6: Validation MAP against training time for AIDS and PTC-FR, for ISONET and GMN-match. ISONET converges faster than GMN-match.

Training scalability analysis. Finally, we compare the scalability of ISONET against its closest competitor, GMN-match. Figure 6 shows the progress of MAP on the validation set as training progresses. We observe that ISONET is at least $5\times$ faster than GMN-match. Training the GS network is much faster than the expensive cross-graph attention network of GMN-match. As a result, ISONET is significantly more efficient to train than GMN-match. This holds even though GMN-match applies the attention layers across *nodes*, whereas, ISONET finds an alignment across *edges*.

5 Conclusion

We have presented ISONET, a new formulation for graph retrieval based on soft subgraph isomorphism. Even in a challenging learning scenario with no direct supervision of node-node correspondence, ISONET performs better than recent, competitive approaches, thanks to a loss objective designed around asymmetric edge coverage, and a novel gadget for edge correspondence search. ISONET also provides plausible explanations for high-scoring response graphs.

It would be of interest to extend such edge alignment techniques to other applications, where absolute subsumption of the query graph is not required. Instead, the most significant common subgraph between the query-corpus pair would dictate similarity. Another useful direction would be the design of interpretable networks for graph edit distance.

References

Bai, Y.; Ding, H.; Bian, S.; Chen, T.; Sun, Y.; and Wang, W. 2019. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 384–392.

Bai, Y.; Ding, H.; Gu, K.; Sun, Y.; and Wang, W. 2020. Learning-based efficient graph similarity computation via multi-scale convolutional set matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3219–3226.

Bernard, F.; Theobalt, C.; and Moeller, M. 2018. Ds*: Tighter lifting-free convex relaxations for quadratic matching problems. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4310–4319.

Cereto-Massagué, A.; Ojeda, M. J.; Valls, C.; Mulero, M.; Garcia-Vallvé, S.; and Pujadas, G. 2015. Molecular fingerprint similarity search in virtual screening. *Methods*, 71: 58–63.

Cho, M.; Lee, J.; and Lee, K. M. 2010. Reweighted random walks for graph matching. In *European conference on Computer vision*, 492–505. Springer.

Cuturi, M. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26: 2292–2300.

Doan, K. D.; Manchanda, S.; Mahapatra, S.; and Reddy, C. K. 2021. Interpretable Graph Similarity Computation via Differentiable Optimal Alignment of Node Embeddings.

Fey, M.; Lenssen, J. E.; Morris, C.; Masci, J.; and Kriege, N. M. 2020. Deep graph matching consensus. *arXiv preprint arXiv:2001.09621*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, 1263–1272. PMLR.

Grohe, M.; Rattan, G.; and Woeginger, G. J. 2018. Graph similarity and approximate isomorphism. *43rd International Symposium on Mathematical Foundations of Computer Science*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035.

Joachims, T. 2002. Optimizing Search Engines Using Clickthrough Data. In *SIGKDD Conference*, 133–142. ACM.

Johnson, J.; Krishna, R.; Stark, M.; Li, L.-J.; Shamma, D.; Bernstein, M.; and Fei-Fei, L. 2015. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3668–3678.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Lai, A.; and Hockenmaier, J. 2017. Learning to predict denotational probabilities for modeling entailment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, 721–730.

Leordeanu, M.; and Hebert, M. 2005. A spectral technique for correspondence problems using pairwise constraints.

Li, Y.; Gu, C.; Dullien, T.; Vinyals, O.; and Kohli, P. 2019. Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning*, 3835–3845. PMLR.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Ling, X.; Wu, L.; Wang, S.; Pan, G.; Ma, T.; Xu, F.; Liu, A. X.; Wu, C.; and Ji, S. 2021. Deep Graph Matching and Searching for Semantic Code Retrieval. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(5): 1–21.

Liu, L.; Hughes, M. C.; Hassoun, S.; and Liu, L.-P. 2021. Stochastic Iterative Graph Matching. *arXiv preprint arXiv:2106.02206*.

Liu, T.-Y. 2009. Learning to Rank for Information Retrieval. In *Foundations and Trends in Information Retrieval*, volume 3, 225–331. Now Publishers.

Lou, Z.; You, J.; Wen, C.; Canedo, A.; Leskovec, J.; et al. 2020. Neural Subgraph Matching. *arXiv preprint arXiv:2007.03092*.

Mena, G.; Belanger, D.; Linderman, S.; and Snoek, J. 2018. Learning latent permutations with gumbel-sinkhorn networks. *arXiv preprint arXiv:1802.08665*.

Morris, C.; Kriege, N. M.; Bause, F.; Kersting, K.; Mutzel, P.; and Neumann, M. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*.

Nowak, A.; Villar, S.; Bandeira, A. S.; and Bruna, J. 2018. Revised note on learning quadratic assignment with graph neural networks. In *2018 IEEE Data Science Workshop (DSW)*, 1–5. IEEE.

Ohlrich, M.; Ebeling, C.; Ginting, E.; and Sather, L. 1993. Subgemini: Identifying subcircuits using a fast subgraph isomorphism algorithm. In *Proceedings of the 30th International Design Automation Conference*, 31–37.

Peng, Y.; Choi, B.; and Xu, J. 2021. Graph Edit Distance Learning via Modeling Optimum Matchings with Constraints.

Qu, J.; Ling, H.; Zhang, C.; Lyu, X.; and Tang, Z. 2021. Adaptive Edge Attention for Graph Matching with Outliers.

Schellewald, C.; and Schnörr, C. 2005. Probabilistic subgraph matching based on convex relaxation. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 171–186. Springer.

Tan, H.-R.; Wang, C.; Wu, S.-T.; Wang, T.-Q.; Zhang, X.-Y.; and Liu, C.-L. 2021. Proxy Graph Matching with Proximal Matching Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 9808–9815.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Vendrov, I.; Kiros, R.; Fidler, S.; and Urtasun, R. 2015. Order-embeddings of images and language. *ICLR 2016*.

Vishwanathan, S. V. N.; Schraudolph, N. N.; Kondor, R.; and Borgwardt, K. M. 2010. Graph kernels. *Journal of Machine Learning Research*, 11: 1201–1242.

Wang, R.; Yan, J.; and Yang, X. 2019. Learning combinatorial embedding networks for deep graph matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3056–3065.

Wong, E. K. 1992. Model matching in robot vision by subgraph isomorphism. *Pattern Recognition*, 25(3): 287–303.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Yan, X.; Yu, P. S.; and Han, J. 2004. Graph indexing: A frequent structure-based approach. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 335–346.

Yu, T.; Wang, R.; Yan, J.; and Li, B. 2019. Learning deep graph matching with channel-independent embedding and hungarian attention. In *International conference on learning representations*.

Yu, T.; Wang, R.; Yan, J.; and Li, B. 2021. Deep Latent Graph Matching. In *International Conference on Machine Learning*, 12187–12197. PMLR.

Yu, T.; Yan, J.; Wang, Y.; Liu, W.; and Li, B. 2018. Generalizing graph matching beyond quadratic assignment model. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 861–871.

Zanfir, A.; and Sminchisescu, C. 2018. Deep learning of graph matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2684–2693.

Zhao, K.; Tu, S.; and Xu, L. 2021. IA-GM: A Deep Bidirectional Learning Method for Graph Matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 3474–3482.